МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ХАРЬКОВСКИЙ НАЦИОНАЛЬНЫЙ УНИВЕРСИТЕТ РАДИОЭЛЕКТРОНИКИ

КАФЕДРА СИСТЕМОТЕХНИКИ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
к лабораторным работам по дисциплине

«INTERNET-ТЕХНОЛОГИИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ИНФОРМАЦИИ»

для студентов специальности

6.050101 – КОМПЬЮТЕРНЫЕ НАУКИ

ЛАБОРАТОРНАЯ РАБОТА № 1
ПРЕДСТАВЛЕНИЕ ДАННЫХ ПРЕДМЕТНОЙ ОБЛАСТИ В ВИДЕ XML-ДОКУМЕНТОВ

Утверждено
на заседании кафедры «Системотехники»
Протокол № 1 от 28 сентября 2017 г.

Харьков 2017

УДК 681.3.07
ББК 32.973.26-018.2.75
I2

I12

Internet-технологии распределенной обработки информации. Методические указания к практическим занятиям для студентов всех форм обучения специальности 6.050101 – «Компьютерные науки» [Электронное издание] / ХНУРЭ; Сост. Ю.В.Мищеряков, А.И.Коваленко, В.М. Решетник – Харьков, 2017. – 43с.

УДК 681.3.07
ББК 32.973.26-018.2.75

СОДЕРЖАНИЕ

1 ПРЕДСТАВЛЕНИЕ ДАННЫХ ПРЕДМЕТНОЙ ОБЛАСТИ В ВИДЕ XML-ДОКУМЕНТОВ

1.1 Цели работы

1. Изучить стандарт языка XML
2. Получить практические навыки по созданию XML-документов, представляющих данные заданной предметной области.
3. Получить практические навыки по созданию пользовательского языка разметки для описания представления данных предметной области с помощью инструкций языка XSD описания XML-схем.
4. Ознакомление с функциональными возможностями интегрированной среды разработки Eclipse.
5. Получить практические навыки создания XML-документов и их XML схем (XSD) в интегрированной среде разработки Eclipse.
6. Получить практические навыки создания таблиц стилей на языке XSLT для отображения созданного XML-документа в браузере.

1.2 Методические указания по организации самостоятельной работы студентов

Во время подготовки к выполнению лабораторной работы необходимо:

- изучить стандарт языка XML;
- изучить стандарт языка XSD описания XML схем;
- ознакомится с интерфейсом и функциональными возможностями интегрированной среды разработки Eclipse;
- изучить порядок использования инструкций языка XML для создания документов, описывающих данные заданной предметной области;
- изучить порядок использования XML-схем (XSD) для генерации и чтения XML-документов;
- изучить порядок использования таблиц стилей на языке XSLT для отображения созданного XML-документа в браузере.

Для подготовки к занятию необходимо использовать конспект лекций [1], методические указания к самостоятельной работе [2] и рекомендованную литературу [3,4] (Эллиот Расти Гарольд, СкоттМинс У. XML. Справочник – главы 1, 2, 4; Хабибуллин И. Разработка WEB-СЛУЖБ средствами Java – глава 1. Обработка документов XML, стр. 14-51).

1.3 Описание лабораторной установки

В качестве лабораторной установки используется персональный компьютер типа IBM PC. Выполнение заданий практического занятия осуществляется с помощью интегрированной среды разработки Eclipse, поддерживающей технологию разработки Java Enterprise Edition (JEE) с соответствующим пакетом Java Development Kit (JDK).

1.4 Порядок выполнения работы и методические указания по ее выполнению

1.4.1 Задание на практическое занятие

В процессе выполнения практического занятия для заданной предметной области необходимо выполнить следующие задания;

– определить назначение распределенного программного приложения и реализуемые бизнес процессы;

– определить структуру данных, передаваемых между частями распределенного приложения, в соответствии с реализуемыми бизнес-процессами;

– создать XML-схему (XSD) для описания структуры данных заданной предметной области;

– создать XML-документ, валидный относительно разработанной XML схемы (XSD);

– описать данные, используя инструкции созданного шаблона XML-документа;

– провести валидацию XML-документа, содержащего данные предметной области, на соответствие созданной XML-схеме (XSD);

– создать таблицу стилей XSLT для отображения созданного XML-документа в браузере.

Перечень тем для выбора предметной области представлен в табл.1.

Таблица 1 – Перечень тем для выбора предметной области

№	Тема
1.	Информационная система «Магазин бытовой техники»
2.	Информационная система «Магазин одежды»
3.	Информационная система «Прокат спортивного инвентаря»

№	Тема
4.	Информационная система «Служба такси»
5.	Информационная система «Центр сервисного обслуживания»
6.	Информационная система «Поликлиника»
7.	Информационная система «Прокат видеофильмов»
8.	Информационная система «Аэропорт» («ЖД вокзал»)
9.	Информационная система «Ресторан»
10.	Информационная система «Кинотеатр»
11.	Информационная система «Автопарк»
12.	Информационная система «Фитнес-Клуб»
13.	Информационная система «Химчистка»
14.	Информационная система «Общежитие»
15.	Информационная система «Отель»
16.	Информационная система «Страховая компания»
17.	Информационная система «Свадебный салон»
18.	Информационная система «Курьерское агентство»
19.	Информационная система «Аптека»
20.	Информационная система «Туристическая фирма»
21.	Информационная система «Абоненты телефонной сети»
22.	Информационная система «Паспортный стол»
23.	Информационная система «Склад»
24.	Информационная система «Расписание занятий»
25.	Информационная система «Планировщик рабочего дня»
26.	Информационная система «Деканат»
27.	Свободно выбранная тема, согласованная с преподавателем

1.4.2 Теоретические сведения

1.4.2.1 Краткие сведения о расширяемом языке разметки (XML)

В 1986 году организацией ISO (International Organization for Standardization) был принят стандарт языка SGML (ISO 8879:1986 Standard Generalized Markup Language – стандартный обобщённый язык разметки). Он позволяет описывать структурированные данные, организовывать и

представлять информацию, содержащуюся в документах, а также обмениваться этой информацией без потери ее «структуры».

На основе SGML по заданию консорциума W3C в 1989 году была разработана спецификация языка HTML (Hyper Text Markup Language – язык разметки гипертекста), а позднее (1996), для расширения функциональности HTML – спецификация языка XML (eXtensible Markup Language – расширяемый язык разметки). Описание спецификаций языка XML находится на сайте W3C по адресу: http://www.w3.org.

Основная цель использования XML – представить текстовую информацию с помощью тегов, структурированных в виде дерева данных. Древовидная структура хорошо описывает бизнес-объекты, конфигурацию, структуры данных и т.п. Данные в таком формате легко могут быть как построены, так и разобраны на любой системе с использованием любой технологии – для этого нужна лишь поддержка работы с текстовыми документами. На основе XML разработаны специализированные стандарты обмена информацией.

1.4.2.2 Понятие «валидных» XML-документов

XML-документ должен иметь строго определенную структуру, задаваемую правилами синтаксиса языка, которые в свою очередь, определяются вариантом используемой спецификации XML.

Корректным (well-formed) XML-документом называют документ, который соответствует спецификации XML. Если документ не соответствует спецификации языка, то он не может считаться XML-документом.

Валидным или действительным (valid) XML-документом называют well-formed документ, соответствующий всем ограничениям, определенным в его XML-схеме. Рассмотрим более подробно термин «XML-схема».

Кроме согласованности со спецификацией, на структуру XML-документа накладываются дополнительные ограничения предметной области, описываемые с помощью XML-схем. XML-схема определяет элементы XML-документа и атрибуты, которые могут быть ассоциированы с элементами. Она также определяет структуру XML-документа, соответствующую древовидной (иерархичной) структуре данных предметной области. В настоящее время широкое распространение получили 3-ри разновидности XML-схем:

DTD-схема (Document Type Definition – определение типа документа) позволяет задать структуру и допустимый набор элементов в XML-документе и

их атрибутов. DTD-схема может быть объявлена как внутри XML-документа, так и вне его. DTD имеет следующие недостатки: синтаксис DTD отличается от синтаксиса XML, что затрудняет его чтение и создание; DTD не поддерживает описание типов данных; DTD не поддерживает пространств имен XML, позволяющих объединять в одном документе элементы документов разной структуры.

XDR-схема (XML-Data Reduced) разработана фирмой Microsoft для парсера собственной версии спецификации языка MS XML. XML-Data – полное имя языка описания схем, предложенного Microsoft консорциуму W3C (1998), а XML-Data Reduced – это часть (reduced) полной рекомендации. Microsoft также разработала и собственную версию Document Content Description (DCD) for XML. XML-схемы в Microsoft Internet Explorer 5.0 и более поздних версиях поддерживают подмножество XML-Data, соответствующее DCD. XDR-схема была объявлена как технология переходного периода к широко используемой в настоящее время XSD-схеме;

XSD-схема (XML Schema Definition – определение схемы XML), которая также часто использует сокращенную аббревиатуру «XML Schema», разработана как альтернатива DTD-схемы. XML Schema лишена недостатков DTD благодаря реализации концепции «пространства имен», что позволяет задавать имена элементов без использования DTD. В языке XML Schema также реализована расширенная система типизации данных.

1.4.2.3 Структура XML-документа

Данные, входящие в документ XML, могут быть описаны в следующих разделах:

– раздел **«XML-декларация»**;

– раздел **«Пролог»**;

– раздел **«Элементы»**;

– раздел **«Атрибуты»**;

– раздел **«Комментарии»**.

1.4.2.4 Декларация XML-документа

Не обязательная XML-декларация обычно находится в первой строке XML-документа. Она состоит из следующих элементов:

– номер версии: <?xml version="1.0"?>. Это обязательный аргумент, если указывается версия "1.1" (версия 1.0 принимается по умолчанию);

– декларация кодировки: <?xml version="1.0" encoding="UTF-8"?>. Это необязательный параметр. Если он используется, то декларация кодировки должна располагаться после значения атрибута version. Декларация кодировки должна содержать значение, представляющее собой используемую кодировку символов (по умолчанию – "UTF-8");

– декларация автономности: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>. Декларация автономности (standalone) используется для задания возможности ("yes"/"no") использования внешнего определения структуры XML-документа с помощью DTD-схем. При указании значения standalone="yes" DTD-схема размещается в теле XML-документа. Декларация автономности необязательна. Значение standalone по умолчанию "yes".

1.4.2.5 Пролог XML-документа

Прологом называются данные, расположенные после открывающего тега документа или после корневого элемента. Он включает сведения, относящиеся к документу в целом – кодировка символов, структура документа, таблицы стилей.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="bookShop.xsl"?>
<!DOCTYPE book SYSTEM "schema.dtd">
<!--Some comments-->
```

В языке XML есть возможность включения в документ инструкций, которые несут определенную информацию для приложений, которые будут его обрабатывать. Инструкции по обработке в XML заключаются в угловые кавычки со знаком вопроса <? содержимое ?>.

1.4.2.6 Элементы XML-документа

Элементы в XML-документе отвечают за хранение данных и являются основными структурными единицами языка XML. Элемент оформляется с помощью парного тега (открывающего <tagName> и закрывающего </tagName>), имя которого произвольно определяется разработчиком и заключается в угловые скобки. Синтаксис элемента с именем тега «tagName»:

<tagName>Содержимое элемента</tagName>

Имена тегам элементов присваиваются по следующим правилам:

– имена тегов могут содержать буквы, цифры и другие символы;

– имена тегов не могут начинаться с цифры или знака препинания;

– имена тегов не могут начинаться с букв «x», «m», «l», «X», «M», «L»;

– в именах тегов не должно быть пробелов (<tag Name> – ошибка);

– нельзя допускать пробелов между открывающейся угловой скобкой и именем тега («< tagName>»,«</ tagName>», «< /tagName>» – ошибка) ;

– имена тегов элементов являются регистрозависимыми (<tagName> и <TagName> – разные теги);

– все элементы должны иметь закрывающий тэг.

В XML элементы могут быть двух типов – пустые и непустые. Пустые элементы не содержат в себе никаких данных, таких как текст или другие конструкции, и могут сокращенно записываться следующим образом: <TagName /> («пустой» тег).

В XML-документе обязательно должен присутствовать единственный корневой элемент, все остальные элементы являются дочерними по отношению к единственному корневому элементу. При этом теги не должны перекрываться.

Пример определения единственного корневого элемента «bookShop» и его дочерних элементов представлен в листинге 1.

Листинг 1 – Пример определения элементов в XML-документе

```
<?xml version="1.0" encoding="UTF-8"?>
<bookShop>
	<book>
		<title>The Emperor's</title>
		<author>Roger</author>
		<ISBN>ISBN-12345-1234</ISBN>
		<price>200</price>
		<category>Monograph</category>
	</book>
	<book>
		<title>New mind</title>
		<author>Penrose</author>
		<ISBN>ISBN-13445-1224</ISBN>
		<price>250</price>
		<category>Monograph</category>
	</book>
</bookShop>
```

1.4.2.7 Атрибуты XML-документа

Теги элементов XML-документа могут иметь атрибуты с присвоенными им значениями, которые помещаются в одинарные или двойные кавычки. Допускается использование одних кавычек внутри других. Имя атрибута определяется произвольно и указывается через пробел после имени тега или предыдущего атрибута. Синтаксис атрибута

<tagName attributeName1="значение" attributeName2='значение'>…</tagName>

Синтаксические правила создания атрибута:

– атрибуты могут определяться в открывающих или пустых тегах, но не в закрывающих тегах;

– количество атрибутов не ограничено;

– несколько атрибутов разделяются пробелами;

– каждое имя атрибута должно быть уникально в рамках одного элемента;

– правила присвоения имен атрибутом имеют те же ограничения, что и имена тегов (см. выше).

Пример определения атрибутов представлен в листинге 2.

Листинг 2 – Пример определения атрибута «id» тега «book»

```
<?xml version="1.0" encoding="UTF-8"?>
<bookShop>
	<book id="1">
		<title>The Emperor's</title>
		<author>Roger</author>
		<ISBN>ISBN-12345-1234</ISBN>
		<price>200</price>
		<category>Monograph</category>
	</book>
	<book id="2">
		<title>New mind</title>
		<author>Penrose</author>
		<ISBN>ISBN-13445-1224</ISBN>
		<price>250</price>
		<category>Monograph</category>
	</book>
</bookShop>
```

1.4.2.8 Комментарии в XML-документе

Содержимое, не предназначенное для синтаксического анализа (например, замечания о структуре документа или редактировании), можно заключить в комментарии. Комментарии начинаются с группы символов «<!--» и заканчиваются группой символов «-->», его синтаксис:

```
<!-- Текст комментария -->
```

Комментарии могут находиться в прологе документа, в том числе в определении типа документа (DTD), после документа и в текстовом содержимом. Комментарии не могут находиться внутри значений атрибутов. Они также не могут находиться внутри тегов.

Синтаксический анализатор считает концом комментария символ «>». После этого символа он возобновляет обработку XML-документа в обычном режиме. Поэтому строка «>» не может находиться внутри комментария. За исключением этого, в комментариях могут использоваться любые допустимые символы XML.

При создании комментария необходимо учитывать следующее:

– в тексте комментария не может быть двух символов «-» подряд.

– комментарий не может заканчиваться символом «-».

1.4.2.9 Текстовые данные XML-документа

Благодаря поддержке набора символов Юникода XML поддерживает целый диапазон символов, в том числе буквы, цифры, знаки препинания и другие символы. Большинство управляющих символов и символы совместимости Юникода не допускаются. Весь текст XML-документа анализируется парсером, т.е. является парсируемым (PCDATA), но в случае необходимости определенные разделы можно «скрыть» для проверки и они будут игнорироваться парсером (т.н. непарсируемые данные, CDATA). Разделы CDATA дают возможность сообщить средству синтаксического анализа, что среди символов, содержащихся в разделе CDATA, отсутствует разметка. Это упрощает создание документов с разделами, в которых могут появиться отдельные символы разметки, но на самом деле разметки нет. В разделы CDATA часто помещают содержимое на языке сценариев, а также образцы содержимого XML и HTML.

Раздел CDATA в схеме документа использует следующую конструкцию:

```
<![CDATA[Some information using <, >,]]>
```

При обнаружении начального тега <![CDATA[, средство синтаксического анализа XML рассматривает все последующее содержимое как символы, не пытаясь интерпретировать их как разметку элемента или сущности. Ссылки на символы в разделах CDATA не работают. Обнаружив завершающий тег]]>, средство синтаксического анализа возобновляет нормальный синтаксический анализ.

Содержимое разделов CDATA может содержать только символы, разрешенные для содержимого XML; нельзя экранировать таким образом управляющие символы и символы совместимости. Кроме того, в раздел CDATA не может входить последовательность]]>, поскольку эти символы означают завершение раздела. Это значит, что разделы CDATA не могут быть вложенными друг в друга. Эта последовательность также используется в некоторых сценариях.

1.4.2.10 Сущности XML-документа

Для использования в XML-документе символов, также как и в HTML, используются экранированные синтаксические конструкции, получившие название сущность (entity). Сущности используются для представления символов, отсутствующих в кодировке XML-документа, или же для представления специальных символов.

В XML сущность – это именованные данные, обычно текстовые, в том числе спецсимволы. В XML существует два метода экранирования специальных символов: ссылка на сущность и ссылка по номеру символа.

Ссылка на сущность (entity references) указывается там, где должна идти данная сущность. Ссылка на сущность состоит из символа амперсанда «&», имени сущности и точки с запятой «;».

Ссылка по номеру символа (numeric character reference) также начинается с амперсанда «&»и заканчивается точкой с запятой «;», но вначале идет символ «#», а далее код символа в кодировке Юникод (в десятичной или шестнадцатеричной системе).

В табл. 1 приведены примеры экранирования предопределенных спецсимволов. В спецификации XML только 5-ть спецсимволов имеют предопределенные ссылки на сущности: «&», «<», «>», «'», и «"».

Например, чтобы текст элемента «NURE&ST», отображался правильно необходимо использовать сущность «&», правильная запись: «NURE&ST».

Таблица 1. Сущности XML

Сущность	Ссылка на сущность	Значение	Ссылка по номеру символа	
lt	**<**	**«<» – меньше чем**	<	
gt	**>**	**«>» – больше чем**	>	
amp	**&**	**«&» – амперсанд**	&	
apos	**'**	**«'» – одиночная кавычка**	'	
quot	**"**	**«"» –двойная кавычка**	"	
-	-	**Неразрывный пробел**		

1.4.2.11 Пространства имен XML-документа

Пространство имён в XML (XML namespace) – это отдельный стандарт, имеющий статус рекомендации, описывающий именованную группу имён тегов элементов и их атрибутов, служащую для обеспечения их уникальности в XML-документе.

Поскольку имена тегов элементов в XML определяются в произвольном порядке (с учетом правил), возможны конфликты, когда в XML-документе используется тег для описания двух различных типов элементов. Для того, чтобы избежать такой ситуации, был предложено использовать пространство имен. Пример конфликта имен приведен в листинге 3.

Листинг 3 – Конфликт тегов <data> по типу назначения элементов

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
	<bookShop>
		<book>
			<title>The Emperor's</title>
			<author>Roger</author>
			<!-- Дата поступления книги на склад -->
			<data>01.08.2017</data >
		</book>
	</bookShop>
	<bookLibrary>
		<book>
			<title>The Emperor's</title>
			<author>Roger</author>
			<!-- Дата выхода книги в свет -->
			<data>10.07.2017</data >
		</book>
```

```
    </bookLibrary>
</books>
```

В приведенном листинге тег элемента «date» описывает дату поступления книги на склад магазина и дату выхода книги в свет.

Пространства имен XML дают возможность избежать конфликта имен тегов элементов. Они задаются при помощи уникальных идентификаторов URI (Uniform Resource Identifier – унифицированный идентификатор ресурса). Синтаксические анализаторы XML не обращаются по этому URI за какой-либо дополнительной информацией, предназначенной для последующей обработки документа. URI – это символьная строка, позволяющая только идентифицировать какой-либо ресурс, в нашем случае, элементы XML-документа. Поэтому URI пространств имен для различных элементов должны быть уникальны. Следует заметить, что в отличие от URI, URL (Uniform Resource Locator – унифицированный локатор ресурса) однозначно идентифицирует и ресурс, и его точное местонахождение.

Для использования пространства имен используется атрибут «xmlns» (xml namespace), значением которого и является URI. Значение URI считается пространством имен заданным по умолчанию. В этом случае все вложенные элементы будут принадлежать пространству имен родительского элемента. При этом не теряется возможность использовать другие пространства имен для дочерних элементов. Пример пространств имен, заданных по умолчанию, приведен в листинге 4.

Листинг 4 – Пример пространств имен, заданных по умолчанию

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <bookShop xmlns="http://nure.ua/bookShop">
        <book>
            <title>The Emperor's</title>
            <author>Roger</author>
            <!-- Дата поступления книги на склад -->
            <data>01.08.2017</data>
        </book>
    </bookShop>
    <bookLibrary xmlns="http://nure.ua/bookLibrary">
        <book>
            <title>The Emperor's</title>
            <author>Roger</author>
```

```
            <!-- Дата выхода книги в свет -->
            <data>10.07.2017</data>
        </book>
    </bookLibrary>
<books>
```

Для упрощения работы с пространствами имен, стандарт предполагает использовать необязательные префиксы. Префикс пространства имен указывается после атрибута «xmlns» и отделяется от него двоеточием (листинг 5).

Листинг 5 – Пример использования префиксов для пространств имен

```
<?xml version="1.0" encoding="UTF-8"?>
<books>
    <Shop:bookShop xmlns:Shop="http://nure.ua/bookShop">
        <Shop:book>
            <Shop:title>The Emperor's</Shop:title>
            <Shop:author>Roger</Shop:author>
            <!--Дата поступления книги на склад--!>
            <Shop:data>01.08.2017</Shop:data >
        </Shop:book>
    </Shop:bookShop>
    <Lib:bookLibrary xmlns:Lib="http://nure.ua/bookLibrary">
        <Lib:book>
            <Lib:title>The Emperor's</Lib:title>
            <Lib:author>Roger</Lib:author>
            <!--Дата выхода книги в свет--!>
            <Lib:data>10.07.2017</Lib:data >
        </Lib:book>
    </Lib:bookLibrary>
</books>
```

Стандарт позволяет объявить несколько пространств имен. Пример такого XML-документа приведен в листинге 6, в котором для описания книги используется одинаковый тег «date», принадлежащий разным пространствам имен.

Листинг 6 – Пример использования нескольких пространств имен

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:Shop="http://nure.ua/bookShop"
```

```
            xmlns:Lib="http://nure.ua/bookLibrary">
      <Shop:book>
            <Shop:title>The Emperor's</Shop:title>
            <Shop:author>Roger</Shop:author>
            <!-- Дата поступления книги на склад -->
            <Shop:data>01.08.2017</Shop:data >
      </Shop:book>
      <Lib:book>
            <Lib:title>The Emperor's</Lib:title>
            <Lib:author>Roger</Lib:author>
            <!-- Дата выхода книги в свет -->
            <Lib:data>10.07.2017</Lib:data >
      </Lib:book>
</books>
```

Можно также использовать комбинацию задания пространства имен по умолчанию и с помощью префиксов. Измененный код предыдущего примера представлен в листинге 7.

Листинг 7 – Использование нескольких пространств имен по умолчанию

```
<?xml version="1.0" encoding="UTF-8"?>
<books xmlns:Shop="http://nure.ua/bookShop"
            xmlns:Lib="http://nure.ua/bookLibrary">
      <book>
            <title>The Emperor's</title>
            <author>Roger</author>
            <!-- Дата поступления книги на склад -->
            <data>01.08.2017</data >
            <!-- Дата выхода книги в свет -->
            <Lib:data>10.07.2017</Lib:data >
      </book>
</books>
```

1.4.2.12 XML Schema Definition (XSD)

XSD-схема (XML Schema Definition – определение схемы XML) или XML Schema – это язык описания структуры XML-документа. Схема включает в себя правила грамматики (или словарь), представляющих собой коллекцию правил, состоящих из определений типов, а также из объявлений элементов и атрибутов. XML Schema используется для определения:

– элементов и их атрибутов XML-документа;

– числа и порядка вхождения дочерних элементов;

– типов данных для элементов и атрибутов;

– значений по умолчанию для элементов и атрибутов.

Язык XML Schema основан на синтаксисе спецификации XML и лишен основных недостатков языка DTD. Одним из главных преимуществ языка XML Schema является возможность описания типов данных. Это позволяет:

– описать допустимый контент для документа;

– валидировать корректность данных;

– накладывать ограничения на данные;

– определять форматы данных;

– конвертировать данные различных типов.

Другим преимуществом языка XML Schema является использование синтаксиса XML. Благодаря этому редактирование файлов XML Schema (.xsd) осуществляется с помощью стандартных редакторов, поддерживающих XML, а для проверки их валидности – стандартный парсер XML.

Рассмотрим пример описания структуры XML-документа с использование XML Schema. Код XML-документа представлен в листинге 8.

Листинг 8 – Код XML-документа с привязкой к схеме (BookShop.xsd)

```
<?xml version="1.0" encoding="UTF-8"?>
<bs:bookShop xmlns:bs="http://nure.ua/bookShop"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://nure.ua/bookShop BookShop.xsd ">
          <bs:book id="0">
     <bs:title>The Pharaoh Contract</bs:title>
     <bs:author> Raymon Huebert Aldridge</bs:author>
     <bs:ISBN>ISBN-97858-5582</bs:ISBN>
     <bs:price>200.0</bs:price>
     <bs:category>Fantasy</bs:category>
     </bs:book>
</bs:bookShop>
```

Для того чтобы использовать ссылку на файл схемы в XML-документа, в коде листинга 8 используются следующие атрибуты:

– bs:bookShop – корневой элемент, для которого указан префикс «bs», определенного для него пространства имен;

– xmlns:bs="http://nure.ua/bookShop" – атрибут xmlns определяет пространство имен «http://nure.ua/bookShop» корневого элемента «bookShop» XML-документа (с префиксом «bs»);

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" – атрибут xmlns определяет пространство имен (по умолчанию) для схемы XML Schema с префиксом «xsi»; Это пространство имен содержит элементы и атрибуты XMLSchema, которые можно включать в документ XML. По общему соглашению префикс xsi используется для этого пространства имен и добавляется в начале имен всех элементов и атрибутов, принадлежащих пространству имен, отделяясь от них двоеточием;

– xsi:schemaLocation="http://nure.ua/bookShop BookShop.xsd" – атрибут schemaLocation с префиксом «xsi»; позволяет указать простанство имен XML-документа, а через пробел путь к файлу и имя файла XML Schema.

В листинге 9 приведен код XML Schema, хранящийся в файле «BookShop.xsd».

Листинг 9 – Код XML Schema (файл BookShop.xsd)

```
<!-- == XML-декларация====================================== -->
<?xml version="1.0" encoding="UTF-8"?>
<!-- == Корневой элемент schema ============================ -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://nure.ua/bookShop"
            xmlns:bs="http://nure.ua/bookShop"
            elementFormDefault="qualified">
<!-- ==Раздел описания элементов============================ -->
<!-- ==Комплексный элемент bookShop ======================== -->
<xsd:element name="bookShop">
      <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="book" type="bs:Book" maxOccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complexType>
</xsd:element>
<!-- ==Комплексный элемент Book ============================ -->
<xsd:complexType name="Book">
      <xsd:complexContent>
            <xsd:extension base="bs:Entity">
                  <xsd:sequence>
                        <xsd:element name="title" type="xsd:string" minOccurs="1"
                                    maxOccurs="1" />
                        <xsd:element name="author" type="xsd:string" minOccurs="1"
                                    maxOccurs="unbounded" />
                        <xsd:element name="ISBN" minOccurs="0" maxOccurs="1">
                              <xsd:simpleType>
                                    <xsd:restriction base="xsd:string">
                              <xsd:pattern value="ISBN-[0-9]{5,5}-[0-9]{4,4}"/>
                                    </xsd:restriction>
                              </xsd:simpleType>
                        </xsd:element>
```

```
                        <xsd:element name="price" type="bs:Price" />
                        <xsd:element name="category" type="bs:Category" />
                  </xsd:sequence>
            </xsd:extension>
            </xsd:complexContent>
      </xsd:complexType>
<!-- == Описания атрибута id ============================== -->
      <xsd:complexType name="Entity" abstract="true">
            <xsd:attribute name="id" type="bs:EntityId" use="required" />
      </xsd:complexType>
<!-- ==Простой элемент Price ============================== -->
      <xsd:simpleType name="Price">
            <xsd:restriction base="xsd:double">
                  <xsd:minInclusive value="0" />
            </xsd:restriction>
      </xsd:simpleType>
<!-- ==Простой элемент Category ============================ -->
      <xsd:simpleType name="Category">
            <xsd:restriction base="xsd:string">
                  <xsd:enumeration value="Fantasy" />
                  <xsd:enumeration value="Action" />
                  <xsd:enumeration value="Love novel" />
                  <xsd:enumeration value="Monograph" />
            </xsd:restriction>
      </xsd:simpleType>
<!-- ==Простой элемент Category ============================ -->
      <xsd:simpleType name="EntityId">
            <xsd:restriction base="xsd:int">
                  <xsd:minInclusive value="-1" />
                  <xsd:maxExclusive value="99999999" />
                  <xsd:pattern value="[0-9]*" />
            </xsd:restriction>
      </xsd:simpleType>
</xsd:schema>
```

Элементы XSD-схемы называют компонентами (components), чтобы отличить их от элементов описываемого документа XML. Корневой компонент схемы носит имя «schema». В спецификации XSD содержимое компонента «schema» определяется следующими разделами:

– «type definitions» – описание применяемых типы данных (простые, комплексные, составные);

– «attribute declarations» – описание глобальных атрибутов;

– «element declarations» – описание элементов, входящих в состав XML-документа;

– «attribute group definitions» – описание используемых групп атрибутов;

– «model group definitions» – описание используемых групп;

– notation declarations – описание дополнительных элементов схемы (шаблоны представления значений);

– «annotations» – аннотации.

Таким образом, вначале определяются применяемые типы данных, как простые, комплексные или составные. Затем следует объявление глобальных атрибутов, которые относятся ко всем элементам сразу. После этого следует раздел объявления элементов, входящих в состав XML-документа. Следом за ними объявляются используемые группы атрибутов. Затем описываются дополнительные элементы схемы, такие как шаблоны значений элементов, или же модели представления. И в самом конце размещаются так называемые аннотации.

В качестве отдельных компонентов схемы применяются 13-ть типов компонентов, разбиваемых обычно на три группы:

– к первой группе относятся определения простых типов, определения составных типов, объявления атрибутов и объявления элементов;

– ко второй группе относятся определения групп атрибутов, определения уникальности элементов, определения моделей представления и так называемых нотаций, то есть элементов, расшифровывающих предназначение того или иного элемента;

– в третью группу входят аннотации, группы моделей представления информации, дополнительные данные о моделях и объявления шаблонов представления содержимого элементов.

Корневым элементом в схеме XML является элемент «schema», который содержит все остальные элементы в документе схемы. В рамках корневого элемента атрибутом «xmlns» определяется пространство имен XMLSchema, которое содержит элементы и атрибуты XSD-схемы.

```
xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

Все элементы XSD начинаются с префикса «xsd:», который указывается для пространства имен XSD, объявленного в корневом элементе экземпляра схемы.

Следующий атрибут «targetNamespace» тега «schema»:

```
targetNamespace="http://nure.ua/bookShop"
```

определяет целевое пространство имен XML-документа, для которого создается XSD-схема.

Атрибут «xmlns» тега «schema»:

```
xmlns:bs="http://nure.ua/bookShop"
```

ссылается на пространство имен с префиксом «:bs» XML-документа, для которого создается XSD-схема.

Атрибут «elementFormDefault» тега «schema»:

elementFormDefault="qualified"

задает значение, должны ли элементы XML-документа, для которого определена XSD-схема, принадлежать пространству имен, указанному в атрибуте targetNamespace. Он может принимать 2-ва значения:

«qualified» – все элементы должны принадлежать целевому пространству имен

«unqualified» – все элементы схемы не принадлежат целевому пространству имен (значение по умолчанию). Для каждого элемента нужно явно указать принадлежность пространству имен, если необходимо.

1.4.2.13 Элементы XML-схем

Описание структуры XML-документа в целом сводится к описанию элементов и их атрибутов. При этом должна учитываться иерархия вхождений элементов друг в друга. При описании схем XML-документов вначале определяются простейшие элементы, которые не содержат дочерних субэлементов. То есть перед описанием некоего родительского элемента сначала описываются все его дочерние субэлементы. Данная концепция естественным образом заимствована из высокоструктурированных языков программирования, в которых нельзя использовать переменные, не объявив их предварительно.

В XSD схеме (см. листинг 9) первая строка содержит XML-декларацию. После декларации всегда обязательно определяется корневой элемент «schema». Далее определяется тип элемента при помощи тэга <EiementType> с обязательным параметром name. В качестве строкового значения этого параметра используется наименование единственного корневого тега XML-документа для которого создается схема. Теги для описания элементов верхнего уровня XML-схем представлены в табл. 2.

Таблица 2. Теги для описания элементов верхнего уровня XML-схем

Элемент	Описание
<xsd:annotation>	**Определяет заметку.**
<xsd:attribute>	**Объявляет атрибут.**

Элемент	Описание
<xsd:attribute>	**Группирует набор объявлений атрибутов таким образом, что их можно включить в качестве группы в определения сложных типов.**
<xsd:complexType>	**Объявляет сложный тип, определяющий набор атрибутов и содержимое элемента.**
<xsd:element>	**Объявляет элемент, который может включать следующие атрибуты: name — обязательный атрибут, определяет имя элемента; type — указывает тип элемента; ref — ссылается на определение элемента, находящегося в другом месте; minOccurs и maxOccurs — количество повторений этого элемента (по умолчанию принимает значение «1»). Чтобы указать, что количество элементов не ограничено, в атрибуте maxOccurs необходимо задать unbounded**
<xsd:group>	**Группирует набор объявлений элементов таким образом, что их можно включить в качестве группы в определения сложных типов**
<xsd:import>	**Определяет пространство имен, на компоненты схемы которого ссылается содержащая схема.**
<xsd:include>	**Включает указанный документ схемы в целевое пространство имен содержащей схемы.**
<xsd:notation>	**Содержит определение нотации, описывающей формат не-XML данных в XML-документе. Определение нотации схемы XML – это видоизменение определений XML 1.0 NOTATION.**
<xsd:redefine>	**Позволяет переопределить в текущей схеме простые и сложные типы, группы и группы атрибутов, полученные из внешних файлов схем.**
<xsd:simpleType>	**Объявление элементов простых типов, которые не имеют атрибутов и дочерних элементов.**

Структура компонента «element» XSD-схемы представлена в листинге 10.

Листинге 10 – Структура компонента «element»

```
<element
abstract = Boolean : false
block = (#all | List of (extension | restriction | substitution))
default = string
final = (#all | List of (extension | restriction))
fixed = string
form = (qualified | unqualified)
id = ID
maxOccurs = (nonNegativeInteger | unbounded) : 1
minOccurs = nonNegativeInteger : 1
name = NCName
```

```
nillable = Boolean : false
ref = QName
substitutionGroup = QName
type = QName
{any attributes with non-schema Namespace}...>
Content: (annotation?, ((simpleType | complexType)?, (unique | key | keyref)*))
</element>
```

Процесс создания XSD-схемы включает в себя два шага – определение и объявление типов элементов или типов атрибутов. Элементы и атрибуты XML-документа объявляются элементами схемы «xsd:element» и «xsd:attribute». Структура же XML-документа определяется элементами схемы «xsd:simpleType» и «xsd:complexType».

Основное объявление элемента состоит из имени и типа данных:

```
<xsd:element name="имя_элемента" type="xsd:тип_данных"/>
```

В XSD- схемах теги (дескрипторы), используемые в XML-документах, разделяются на две категории типов сложные и простые. Элементы сложных типов могут содержать другие элементы, а также обладают определенными атрибутами; элементы простых типов такими возможностями не обладают.

1.4.2.14 Простые (simple) типы XML-схем

Элементы, которые не имеют атрибутов и дочерних элементов, называются простыми и должны иметь простой тип данных.

Теги для описания простых типов элементов XML-схем представлены в табл. 3.

Таблица 3. Теги для описания простых типов XSD-схем

Элемент	Описание
<xsd:annotation>	**Определяет аннотацию.**
<xsd:appinfo>	**Задает сведения, используемые приложениями в элементе «annotation»**
<xsd:documentation>	**Задает сведения, которые читают или используют пользователи в элементе «annotation»**
<xsd:element>	**Объявляет элемент.**
<xsd:list>	**Определяет коллекцию из одного определения simpleType**
<xsd:restriction> (simpleType)	**Задает ограничения на определение simpleType.**

Элемент	Описание
<xsd:union>	**Определяет коллекцию из нескольких определений simpleType.**

Синтаксис простого элемента:

```
<xsd:element name="nnn" type="ttt"/>
```

где «"nnn"» – имя элемента, а «"ttt"» – его тип.

Есть две главных категории простых типов: встроенные типы и определенные пользователем простые типы.

Язык XSD имеет большое количество встроенных простых типов данных. Встроенные типы включают в себя примитивные типы и производные. Примитивные типы данных не получены из других типов данных. Например, числа с плавающей запятой – математическое понятие, которое не получено из других типов данных. Производные типы данных определены в терминах существующих типов данных. Например, целое число – частный случай, полученный из десятичного типа данных.

В табл. 4 представлен список примитивных типов данных XSD-схемы, которые могут быть применены к типу данных и для описания типа данных.

Таблица 4 – Примитивные типы данных XSD-схем

Тип данных	Ограничения (аспекты, facets)	Описание
string	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет символьную строку.**
Boolean	**pattern, whiteSpace**	**Представляет логическое значение, которое может быть true или false.**
decimal	**enumeration, pattern, totalDigits, fractionDigits, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет произвольное число.**
float	**pattern, enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет 32-битовое число с плавающей запятой одиночной точности.**

Тип данных	Ограничения (аспекты, facets)	Описание
double	**pattern, enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет 64-битовое число с плавающей запятой двойной точности.**
duration	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет продолжительность времени. Шаблон для duration следующий - PnYnMnDTnHnMnS, где nY представляет число лет; nM - месяцев; nD - дней; T - разделитель даты и времени; nH - число часов; nM - минут; nS - секунд.**
dateTime	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет конкретное время. Шаблон для dateTime следующий - CCYY-MM-DDThh:mm:ss, где CC представляет столетие; YY - год; MM - месяц; DD - день; T - разделитель даты и времени; hh - число часов; mm - минут; ss - секунд. При необходимости можно указывать доли секунды. Например, сотые доли в шаблоне: ss.ss**
time	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет конкретное время дня. Шаблон для time следующий -hh:mm:ss.sss (долевая часть секунд необязательна).**
date	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет календарную дату. Шаблон для date такой - CCYY-MM-DD (здесь необязательна часть, представляющая время).**
gYearMonth	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет конкретный месяц конкретного года (CCYY-MM).**
gYear	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет календарный год (CCYY).**

Тип данных	Ограничения (аспекты, facets)	Описание
gMonthDay	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет конкретный день конкретного месяца (--MM-DD).**
gDay	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет календарный день (---DD).**
gMonth	**enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace**	**Представляет календарный месяц (--MM--).**
hexBinary	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет произвольную шестнадцатерично-закодированную двоичную информацию. HexBinary - набор двоичных октетов фиксированной длины, состоящий из четырех пар шестнадцатеоисных символов. Например, 0-9a-fA-F.**
base64Binary	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет произвольную Base64-закодированную двоичную информацию. Base64Binary - набор двоичных октетов фиксированной длины.**
anyURI	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет URI как определено в RFC 2396. Значение anyURI может быть абсолютно или относительно, и может иметь необязательный идентификатор фрагмента.**
QName	**length, enumeration, pattern, maxLength, minLength, whiteSpace**	**Представляет составное имя. Имя составлено из префикса и локального названия, отделенного двоеточием. И префикс и локальные названия должны быть NCNAME. Префикс должен быть связан с namespace URI ссылкой, используя объявление пространства имени.**
NOTATION	**length, enumeration, pattern, maxLength, minLength, whiteSpace**	**Представляет тип атрибута СИСТЕМЫ ОБОЗНАЧЕНИЙ. Набор QNAMES.**

В табл.5 приведена XSD-схема, содержащая определение простых типов.

Таблица 5 – Использование примитивных типов данных

XML-документ	XSD-схема

<lastname>Refsnes</lastname> <age>36</age> <dateborn>1970-03-27</dateborn>	<xsd:element name="lastname" type="xsd:string"/> <xsd:element name="age" type="xsd:integer"/> <xsd:element name="dateborn" type="xsd:date"/>

В табл.6 представлен список производных типов данных XSD-схемы и ограничители, которые могут быть применены к типу данных и в описании типа данных.

Таблица 6 – Производные типы данных XSD-схемы

Тип данных	Ограничения (аспекты, facets)	Описание
normalizedString	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет нормализованные строки. Этот тип данных получен из string.**
token	**enumeration, pattern, length, minLength, maxLength, whiteSpace**	**Представляет маркированные строки. Этот тип данных получен из normalizedString.**
language	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет идентификаторы естественного языка (определенный RFC 1766). Этот тип данных получен из token**
IDREFS	**length, maxLength, minLength, enumeration, whiteSpace**	**Представляет тип атрибута IDREFS. Содержит набор значений типа IDREF.**
ENTITIES	**length, maxLength, minLength, enumeration, whiteSpace**	**Представляет тип атрибута ENTITIES. Содержит набор значений типа ENTITY.**
NMTOKEN	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет тип атрибута NMTOKEN. NMTOKEN - набор символов имен (символы, цифры и другие символы) в любой комбинации. В отличие отName и NCNAME, NMTOKEN не имеет никаких ограничений на первый символ. Этот тип данных получен из token.**
NMTOKENS	**length, maxLength, minLength, enumeration, whiteSpace**	**Представляет тип атрибута NMTOKENS. Содержит набор значений типа NMTOKEN.**
Name	**length, pattern, maxLength, minLength,**	**Представляет имена в XML. Name - лексема(маркер), которая начинается с символа, символа подчеркивания или**

Тип данных	Ограничения (аспекты, facets)	Описание
	enumeration, whiteSpace	**двоеточия и продолжается символами имен (символы, цифры, и другие символы). Этот тип данных получен из token.**
NCName	**length, pattern, maxLength, minLength, enumeration, whiteSpace**	**Представляет неколонкированные названия. Этот тип данных - тот же, что и Name, но не может начинаться с двоеточия. Этот тип данных получен из Name.**
ID	**length, enumeration, pattern, maxLength, minLength, whiteSpace**	**Представляет тип атрибута ID, определенный в XML 1.0 Рекомендации. ИДЕНТИФИКАТОР не должен иметь двоеточия (NCName) и должен быть уникален в пределах XML документа. Этот тип данных получен из NCNAME.**
IDREF	**length, enumeration, pattern, maxLength, minLength, whiteSpace**	**Представляет ссылку к элементу, имеющему атрибут ID, который точно соответствует установленному ИДЕНТИФИКАТОРУ. IDREF должен быть NCNAME и должен быть значением элемента или атрибута типа ID в пределах XML документа. Этот тип данных получен из NCNAME.**
ENTITY	**length, enumeration, pattern, maxLength, minLength, whiteSpace**	**Представляет тип атрибута ENTITY. Это - ссылка к неанализируемому объекту с именем, которое точно соответствует установленному имени. ENTITY должен быть NCNAME и должен быть объявлен в схеме как неанализируемое имя объекта. Этот тип данных получен из NCNAME.**
integer	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет последовательность десятичных цифр с необязательным знаком (+ или -). Этот тип данных получен из decimal.**
nonPositiveInteger	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число, меньшее или равное нулю. NonPositiveInteger состоит из отрицательного знака (-) и последовательности десятичных цифр. Этот тип данных получен из целого числа.**
negativeInteger	**enumeration, fractionDigits, pattern, minInclusive, minExclusive,**	**Представляет целое число, меньшее нуля. Этот тип данных получен из nonPositiveInteger.**

Тип данных	Ограничения (аспекты, facets)	Описание
	maxInclusive, maxExclusive, totalDigits, whiteSpace	
long	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимальным значением -9223372036854775808 и максимумом 9223372036854775807. Этот тип данных получен из целого числа.**
int	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимальным значением -2147483648 и максимумом 2147483647. Этот тип данных получен из long.**
short	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимальным значением -32768 и максимумом 32767. Этот тип данных получен из int.**
byte	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимальным значением -128 и максимумом 127. Этот тип данных получен из short.**
nonNegativeInteger	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число, большее равное нулю. Этот тип данных получен из целого числа.**
unsignedLong	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive,**	**Представляет целое число с минимумом нуль и максимумом 18446744073709551615. Этот тип данных получен из nonNegativeInteger.**

Тип данных	Ограничения (аспекты, facets)	Описание
	maxExclusive, totalDigits, whiteSpace	
unsignedInt	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимумом нуль и максимумом 4294967295. Этот тип данных получен из unsignedLong.**
unsignedShort	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимумом нуль и максимумом 65535. Этот тип данных получен из unsignedInt.**
unsignedByte	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число с минимумом нуля и максимума 255. Этот тип данных получен из unsignedShort.**
positiveInteger	**enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace**	**Представляет целое число, которое является большим чем нуль. Этот тип данных получен из nonNegativeInteger.**

Именованные ограничения, называемыми аспектами (facets), ограничивают допустимые значения простых типов. Синтаксис применения аспектов следующий:

```
<xsd:restriction base="тип_данных">
     <xsd:имя_аспекта value="значение_аспекта"/>
</xsd:restriction>
```

Для указания основного типа используется элемент «restriction» (определение). Его атрибут «base» указывает на основной тип. В элемент «restriction» можно включить ряд ограничений на значения типа (табл. 7).

Таблица 7 –Типы разрешенных ограничений, в зависимости от типа

Ограничения	Описание	Пример
xsd:enumeration	**Определяет список допустимых значений**	**<xsd:element name="car"> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:enumeration value="Audi"/> <xsd:enumeration value="Golf"/> <xsd:enumeration value="BMW"/> </xsd:restriction> </xsd:simpleType> </xsd:element>**
xsd:fractionDigits	**Определяет максимальное число цифр после запятой. Должно быть больше либо равно 0.**	
xsd:maxExclusive	**Определяет верхнюю границу для цифровых величин (значение должно быть меньше указанной величины)**	
xsd:minExclusive	**Определяет минимальное значение для цифровых значений (значение должно быть больше указанной величины)**	
xsd:maxInclusive	**Определяет верхнюю границу для цифровых величин (значение должно быть меньше либо равно указанной величины)**	**<xsd:element name="age"> <xsd:simpleType> <xsd:restriction base="xsd:integer"> <xsd:minInclusive value="0"/> <xsd:maxInclusive value="120"/> </xsd:restriction> </xsd:simpleType> </xsd:element>**
xsd:minInclusive	**Определяет наименьшее разрешенное значение для цифровых величин (значение должно быть больше либо равно указанной величине**	
xsd:length	**Определяет точное количесто символов или элементов в списке. Должно быть больше либо равно 0.**	**<xsd:element name="password"> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:length value="8"/> </xsd:restriction> </xsd:simpleType> </xsd:element>**
xsd:maxLength	**Определяет максимальное число символов в значении или элементов в вписке. Должно быть больше либо равно 0.**	**xsd:element name="password"> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:minLength value="5"/> <xsd:maxLength value="8"/> </xsd:restriction> </xsd:simpleType> </xsd:element>**
xsd:minLength	**Определяет минимальное количиство символов или элементов в списке. Должно быть больше либо равно 0.**	

xsd:pattern	**Определяет регулярное выражение, определяющее разрешенные значения**	
xsd:totalDigits	**Определяет точное количество цифр в цифровой величине. Должно быть больше 0.**	
xsd:whiteSpace	**Определяет как обрабатываются символы разделители (переводы строки, табуляции, пробелы, возвраты каретки). value="collapse"|"preserve"|"replace"**	**<xsd:element name="address"> <xsd:simpleType> <xsd:restriction base="xsd:string"> <xsd:whiteSpace value="collapse"/> </xsd:restriction> </xsd:simpleType> </xsd:element>**
	value="preserve"	**Никакая нормализация не выполняется.**
	value="replace"	**Все #x9 (tab), #xA (line feed) and #xD (carriage return) заменяются на #x20 (пробел).**
	value="collapse"	**После replace-обработки все внутренние цепочки #x20 разрушаются до одного пробела, а окружающие пробелы удаляются.**

Аспекты могут быть указаны только однажды в определении типа, кроме «enumeration» и «pattern» – они могут иметь многократные вхождения и группируются.

В языке XSD реализована концепция именованных типов. Например, при создании определения, можно присвоить этому определению имя, чтобы повторно использовать его в XSD-схеме.

Рассмотрим пример создания элемента простого типа (simpleType) с именем «txt15pre». В результате будет создано именованное ограничение. После этого появляется возможность применять это ограничение и к другим элементам в схеме. Это полезно, когда в определении применяются аспекты ограничения типа данных, чтобы не повторять их каждый раз в других определениях. Например, элемент «simpleType» может быть связан с элементом «Фамилия» и атрибутом «Телефон» для объявления содержания этих элемента и значения атрибута как строковых данных:

```
<xsd:simpleType name="txt15pre">
     <xsd:restriction base="xsd:string">
          <xsd:maxLength value="15"/>
          <xsd:whiteSpace value="preserve"/>
     </xsd:restriction>
</xsd:simpleType>
```

```
<xsd:element name="Фамилия" type="txt15pre"/>
<xsd:attribute name="Телефон" type="txt15pre" use="required"/>
```

Значение атрибута use="required" означает обязательность использования объявленного атрибута. Другими предопределенными значениями атрибута «use» элемента схемы «xsd:attribute» могут быть ключевые слова «optional» и «prohibited». Первое из них означает необязательность использования, а второе – запрещает использование объявленного атрибута. Такая необходимость возникает в случае локального объявления ранее определенной группы атрибутов элементом схемы «xsd:attributeGroup», например:

```
<xsd:attributeGroup name="Связь">
       <xsd:attribute name="Телефон" type="txt15pre"/>
       <xsd:attribute name="Факс" type="txt15pre"/>
</xsd:attributeGroup>
```

Далее в контексте определения элемента сложного типа делается ограничение на применение атрибутов этой группы:

```
<xsd:complexType name="Клиент">
     <xsd:complexContent>
          <xsd:restriction base="xsd:Связь">
               <xsd:attribute name="Телефон" use="required"/>
               <xsd:attribute name="Факс" use="prohibited"/>
          </xsd:restriction>
     </xsd:complexContent>
</xsd:complexType
```

1.4.2.15 Комплексные (complex) типы элементов XML-схем

Комплексные (сложные) типы элементов XML-схем – это элементы, содержащие вложенные элементы или атрибуты.

Сложные элементы создаются с помощью элемента «complexType». Так же, как и в простом типе атрибут «name» задает имя типа. Для указания, что элементы внутри описываемого сложного типа должны располагаться в определенной последовательности, используются элементы «sequence», «all», «choice». Он может содержать элементы «element», определяющие содержание сложного типа.

Если тип может содержать не только элементы, но и текстовую информацию, необходимо задать значение атрибута «mixed="true"». Кроме элементов, тип может содержать атрибуты, которые создаются элементом «attribute». Атрибуты элемента «attribute»: «name» – имя атрибута, «type» – тип

значения атрибута. Для указания, обязан ли использоваться атрибут, нужно использовать атрибут «use», который принимает значения «required», «optional», «prohibited». Для установки значения по умолчанию используется атрибут «default», а для фиксированного значения – атрибут «fixed».

В табл. 8 приведены элементы, создающие определения комплексных (сложных) типов.

Таблица 8. Определения сложных типов XSD-схем

Элемент	Описание
<xsd:all>	**Позволяет элементам группы появляться (или не появляться) в содержащем элементе в любом порядке.**
<xsd:annotation>	**Определяет заметку.**
<xsd:any>	**Разрешает любому элементу из указанных пространств имен появляться в содержащем их элементе sequence или choice.**
<xsd:anyAttribute>	**Разрешает любому атрибуту из указанных пространств имен появляться в содержащем их элементе complexType или attributeGroup.**
<xsd:appinfo>	**Задает сведения, используемые приложениями в элементе annotation.**
<xsd:attribute>	**Объявляет атрибут.**
<xsd:attributc>	**Группирует набор объявлений атрибутов таким образом, что их можно включить в качестве группы в определения сложных типов.**
<xsd:choice>	**Позволяет присутствовать в элементе-контейнере одному и только одному элементу выбранной группы.**
<xsd:complexContent>	**Содержит расширения или ограничения для сложного типа, хранящего смешанное содержимое или только элементы.**
<xsd:documentation>	**Задает сведения, которые читают или используют пользователи в элементе annotation.**
<xsd:element>	**Объявляет элемент.**
<xsd:extension> (simpleContent)	**Содержит расширения simpleContent. Выполняется расширение простого или сложного типа, содержащего простое содержимое, путем добавления указанных атрибутов, групп атрибутов, либо атрибута anyAttribute.**
<xsd:extension> (complexContent)	**Содержит расширения для complexContent**
<xsd:group>	**Группирует набор объявлений элементов таким образом, что их можно включить в качестве группы в определения сложных типов**
<xsd:restriction> (simpleContent)	**Задает ограничения на определение simpleContent.**

Элемент	Описание
<xsd:restriction> (complexContent)	**Задает ограничения на определение complexContent.**

Модель содержания элемента сложного типа – формальное описание структуры и допустимого содержания элемента, которое используется для проверки правильности XML-документа. Модель содержания может ограничивать документ до некоторого набора элементных типов и атрибутов, описывать и поддерживать связи между этими различными компонентами и уникально обозначать отдельные элементы. Свободное использование модели содержания позволяет разработчикам изменять структурную информацию.

Перечень объявлений дочерних элементов приводится в структуре группирующих XSD-элементов «choice», «sequence», и «all».

Элемент «xsd:choice» позволяет только одному из элементов, содержащихся в группе присутствовать в составе элемента XML-документа. Элемент «xsd:sequence» требует появления элементов группы в точно установленной последовательности в составе элемента XML-документа. Элемент «xsd:all» позволяет субэлементам в группе быть (или не быть) в любом порядке в составе элемента XML-документа.

Элемент «xsd:group» используется для четкого определения группы и для ссылки к именованной группе. Использование модели группы позволяет определить набор элементов, которые могут быть повторены в XML-документе. Это полезно для формирования определения комплексного типа. Именованную модель группы можно далее определить, используя «xsd:sequence», «xsd:choice» или «xsd:all» в дочерних элементах. Именованные группы должны определяться в корне схемы. При необходимости многократного использования перечня элементов, определенного в группе достаточно дать ссылку на именованную группу «xsd:group ref="имя_группы"»

Определения сложных типов создаются с использованием элемента «complexType», его атрибутов и любых допустимых аспектов (ограничений). Обычно, сложные типы содержат набор объявлений элементов, объявлений атрибутов и элементных ссылок.

```
<xsd:element name="имя_элемента" type="xsd:тип_данных">
     <xsd:complexType>
          <xsd:sequence>
               <xsd:element name="имя_элемента" type="xsd:тип_данных"/>
          </xsd:sequence>
     <xsd:attribute name="имя_атрибута" type="xsd:тип_данных"/>
```

```
    </xsd:complexType>
</xsd:element>
```

В табл. 9 приведен пример определения XSD-схемы для сложного типа.

Таблица 9 – Пример определения сложных типов XSD-схем

XML-документ	XSD-схема
<employee> <firstname>John</firstname> <lastname>Smith</lastname> </employee>	**<xsd:element name="employee"> <xsd:complexType> <xsd:sequence> <xsd:element name="firstname" type="xsd:string"/> <xsd:element name="lastname" type="xsd:string"/> </xsd:sequence> </xsd:complexType> </xsd:element>**
<employee> <firstname>John</firstname> <lastname>Smith</lastname> </employee>	**<xsd:element name="employee" type="personinfo"/> <xsd:complexType name="personinfo"> <xsd:sequence> <xsd:element name="firstname" type="xsd:string"/> <xsd:element name="lastname" type="xsd:string"/> </xsd:sequence> </xsd:complexType>**
<?xml version="1.0"?> <BOOK InStock="true"> <TITLE>title 1</TITLE> <AUTHOR>author 1</AUTHOR> <BINDING>trade paperback</BINDING> <PAGES>473</PAGES> <PRICE>10.95</PRICE> </BOOK>	**<?xml version="1.0"?> <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"> <xsd:element name="BOOK"> <xsd:complexType> <xsd:sequence> <xsd:element name="TITLE" type="xsd:string"/> <xsd:element name="AUTHOR" type="xsd:string"/> <xsd:element name="BINDING" type="xsd:string"/> <xsd:element name="PAGES" type="xsd:positiveInteger"/> <xsd:element name="PRICE" type="xsd:decimal"/> </xsd:sequence> <xsd:attribute name="InStock" type="xsd:boolean" use="required"/> </xsd:complexType> </xsd:element> </xsd:schema>**

1.4.2.16 Атрибуты XML-схем

В табл. 10 приведены элементы, определяющие атрибуты в схемах.

Таблица 10 – Элементы, определяющие атрибуты XML-схем

Элемент	**Описание**
<xsd:anyAttribute>	**Разрешает любому элементу из указанных пространств имен появляться в содержащем их элементе complexType или attributeGroup.**
<xsd:attribute>	**Объявляет атрибут.**
<xsd:attribute>	**Группирует набор объявлений атрибутов таким образом, что их можно включить в качестве группы в определения сложных типов.**

Все атрибуты декларируются, как простые типы. Простые элементы не могут иметь атрибуты. Если у элемента есть атрибуты, то он относится к комплексным или составным типам. Но сам по себе атрибут всегда декларируется, как простой тип.

Атрибут декларируется следующим образом:

```
<xsd:attribute name="xxx" type="yyy"/>
```

где «xxx» – имя атрибута, а «yyy» – тип данных атрибута.

Типы данных, используемые в атрибутах, совпадают с декларациями простых типов и рассмотрены выше.

Пример описания атрибута приведен в табл. 11.

Таблица 11 –Пример описания атрибутов

XML-документ	`<lastname lang="EN">Smith</lastname>`
XSD-схема	`<xsd:attribute name="lang" type="xsd:string"/>`

Атрибуты могут иметь значения по умолчанию или фиксированные значения. Значение по умолчанию присваивается атрибуту автоматически, если не определено никакого другого значения.

Пример объявления значения по умолчанию ("EN"):

```
<xsd:attribute name="lang" type="xsd:string" default="EN"/>
```

Фиксированное значение также присваивается атрибуту автоматически, но при этом вы не можете определить никакого другого значения.

Пример объявления фиксированного значения ("EN"):

```
<xsd:attribute name="lang" type="xsd:string" fixed="EN"/>
```

По умолчанию атрибуты являются необязательными для использования. Чтобы декларировать обязательный атрибут, следует воспользоваться атрибутом «use»:

```
<xsd:attribute name="lang" type="xsd:string" use="required"/>
```

Если XML элемент или атрибут имеет определение типа данных, то это накладывает ограничение по контенту этого элемента или атрибута. Например, если XML-элемент имеет тип «xsd:date» и содержит строку, например, «Hello World», то этот элемент не пройдет валидацию или проверку на корректность данных.

1.4.3 Порядок выполнения работы

Задания практического занятия необходимо выполнять поэтапно. Примерная последовательность этапов работы:

1. Выбрать тему, определяющую предметную область (табл.1).
2. Определить назначение распределенного программного приложения. Составить перечень реализуемых бизнес процессов.
3. Определить структуру и типы данных, передаваемых между частями распределенного приложения, в соответствии с реализуемыми бизнес-процессами.
4. С использованием IDE Eclipse создать проект, включающий XML-документ. Провести валидацию XML-документа.
5. С использованием IDE Eclipse создать XSD-схему XML-документа. Провести валидацию XSD-схемы.
6. С использованием IDE Eclipse сгенерировать XML-документ на основе XSD-схемы. Произвести сравнение сгенерированного и созданного в пп.4 XML-документов. При необходимости произвести корректировку XSD-схемы. и повторить процесс сравнения.
7. Заполнить сгенерированный и валидный XML-документ данными.
8. Создать таблицу стилей на языке XSLT для отображения созданного XML-документа в браузере.
9. Оформить отчет по проделанной работе.

1.5 Содержание отчета

Отчет должен содержать:

- цель работы;
- постановку задачи;
- описание предметной области и перечень реализуемых бизнес-процессов;
- код файла с XSD-схемой;

- код сгенерированного на основе XSD-схемы XML-файла, с внесенными данными;
- результаты валидации XSD-схемы и XML-файла;
- код файла с таблицей стилей на языке XSLT и скриншоты отображения созданного XML-документа в браузере;
- выводы по работе.

Критерии оценки для оформленного отчета (без теории)

1. Создан корректный (well-formed) и валидный (valid) XML-документ;
2. Создана корректная (well-formed) и валидная (valid) XSD-схема XML-документа;
3. В XSD-схеме при описании элементов XML-документа:

– определено пространство имен и его префиксы;

– описаны все элементы XML-документа, с указанием типа, ограничений (minOccurs, maxOccurs, minInclusive, minExclusive, maxInclusive, maxExclusive, length, maxLength, minLength, fractionDigits, totalDigits), регулярных выражений (<xsd:pattern>), а также с использованием и без использования <xsd:restriction>;

– при описании компонентов сложных типов использованы все 3-ри инструкции: <xsd:all>, <xsd:choice>, <xsd:sequence>;

– должен быть описан компонент (<xsd:element>), содержащий предопределенный список (<xsd:enumeration>) возможных принимаемых значений;

– описан обязательный атрибут, его тип и ограничения на значения;

– описан необязательный атрибут, его тип и ограничения на значения;

– созданная таблица стилей XSLT позволяет отобразить XML-документ в браузере. При этом используются различные форматы данных (шрифт: размер шрифта, начертание; таблицы, параграфы, списки; реализована фильтрация данных).

1.6 Контрольные вопросы и задания

1. Что могут содержать сущности, из которых состоит XML-документ?
2. Для чего в XML-документах используются теги?
3. Что идет перед корневым элементом XML-документа?
4. Что такое начальный и конечный теги?

5. Как можно записать пустой элемент?

6. Что может быть именем элемента или атрибута в XML-документе?

7. Какой XML-документ является корректным.

8. Какой XML-документ является валидным.

9. Как в XML-документе обозначаются комментарии?

10. Допустимы ли в XML-документе не закрытые теги?

11. Что такое перекрывающиеся теги и допустимы ли они в XML-документах?

12. Что такое спецсимволы в XML-документах?

13. Как можно экранировать спецсимволы в XML-документах?

14. Что такое ссылка на сущность?

15. Сколько предопределенных сущностей есть в языке XML?

16. Какие встроенные типы языка описания XML-схем используются в настоящее время?

17. Что понимается под XML Schema?

18. Для чего используется XML Schema?

19. Каковы преимущества XML Schema перед DTD?

20. Как подключить XML Schema к XML-документу?

21. Каков синтаксис объявления простого элемента в XML Schema?

22. Какие типы данных существуют в XML Schema?

23. Описать простой тип, который основывается на positiveInteger. Допустимые значения для данного типа лежат в диапазоне [771, 116].

24. Описать простой тип, для которого допустимым значением будет строка из 5 букв в нижнем регистре.

25. Описать простой тип, для которого допустимым будет одно из перечисленных значение: Caption1, Caption2, Caption3.

26. Приведите XSD-схему для следующего элемента

```
<book id=”5465464” name=”BookTitle” pages=”450”/>
```

где id и name – обязательные атрибуты, pages – необязательный.

27. Приведите XSD-схему для следующего элемента

```
<result unit=”cm” precision=”2” > 121.58</result>
```

28. Какие model group выделяют для описания вложенных элементов. Чем они отличаются (<sequence>, <all>, <choice>).

29. Приведите XSD-схему для следующего элемента

```
<person id=”4546464”>
```

```
    <name>Ivan</name>
    <surname>Ivanov</surname>
    <birthday>01.01.1992</birthday>
    <phone>0953682547</phone>
    <phone>0679871236</phone>
</person>
```

30. Каким образом в XSD реализован механизм наследования? Приведите пример.
31. Назначение языка XSL.
32. Назначение языка XSLT.

Учебное издание / Электронное учебное издание

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для практических занятий по дисциплине
«INTERNET-ТЕХНОЛОГИИ РАСПРЕДЕЛЕННОЙ ОБРАБОТКИ ИНФОРМАЦИИ»
для студентов специальности
6.050101 – КОМПЬЮТЕРНЫЕ НАУКИ

Составители: МИЩЕРЯКОВ Юрий Валентинович
КОВАЛЕНКО Андрей Иванович
РЕШЕТНИК Виктор Михайлович

Ответственный выпускающий: Гребенник И.В.
Редактор: Мищеряков Ю.В.
Компьютерная верстка: Коваленко А.И.

План 2017 (первое полугодие), поз. 9
Подп. к печ. 30.01.2017. Формат 60x84 1/16. Способ печати – ризография
Усл. печ. лист. 9,5. Учет. изд.лист. 8,4 Тираж 50 экз.
Цена договорная Зам. №1-9

ХНУРЭ. Украина. 61166, Харьков, просп. Науки, 14

Отпечатано в учебно-научном
издательско-полиграфическом центре ХНУРЭ
61166, Харьков, просп. Науки, 14