МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФЕДРА СИСТЕМОТЕХНІКИ

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи № 1
з дисципліни
«INTERNET ТЕХНОЛОГІЇ»

Лабораторна робота № 1
РОЗРОБКА СПЕЦИФІКАЦІЙ ВЗАЄМОДІЇ ПІДСИСТЕМ РОЗПОДІЛЕНОГО ПРОГРАМНОГО ЗАСТОСУВАННЯ

Затверджене
на засіданні кафедри «Системотехніки»
Протокол № 1 від 30 вересня 2020 р.

Харків 2020

УДК 681.3.07
ББК 32.973.26-018.2.75
І2

І12

Internet технології. Методичні вказівки до виконання лабораторної роботи № 1 з дисципліни «Internet технології» для студентів усіх форм навчання спеціальності 151 – «Автоматизація та комп'ютерно-інтегровані технології» / Упоряд. : Ю.В.Міщеряков, А.І.Коваленко, В.М. Решетник –Харків, 2020. – 40с.

УДК 681.3.07
ББК 32.973.26-018.2.75

ЗМІСТ

1 РОЗРОБКА СПЕЦИФІКАЦІЙ ВЗАЄМОДІЇ ПІДСИСТЕМ РОЗПОДІЛЕНОГО ПРОГРАМНОГО ЗАСТОСУВАННЯ

1.1 Мета роботи

1. Вивчити стандарт мови XML
2. Одержати практичні навички по створенню XML-документів, що представляють дані заданої предметної області.
3. Одержати практичні навички по створенню користувацької мови розмітки для опису та поданню даних предметної області за допомогою інструкцій мови XSD опису XML-схем.
4. Ознайомлення з функціональними можливостями інтегрованого середовища розробки Eclipse.
5. Одержати практичні навички створення XML-документів і їх XML схем (XSD) в інтегрованому середовищі розробки Eclipse.

1.2 Методичні вказівки по організації самостійної роботи студентів

Під час підготовки до виконання лабораторної роботи необхідно:

- вивчити стандарт мови XML;
- вивчити стандарт мови XSD опису XML схем;
- ознайомиться з інтерфейсом і функціональними можливостями інтегрованого середовища розробки Eclipse;
- вивчити порядок використання інструкцій мови XML для створення документів, що описують дані заданої предметної області;
- вивчити порядок використання XML-схем (XSD) для генерації й читання XML- документів.

1.3 Опис лабораторної установки

Як лабораторна установка використовується персональний комп'ютер типу IBM PC. Виконання завдань лабораторної роботи здійснюється за допомогою інтегрованого середовища розробки Eclipse, що підтримує технологію розробки Java Enterprise Edition (JEE) з відповідним пакетом Java Development Kit (JDK).

1.4 Порядок виконання роботи й методичні вказівки по її виконанню

1.4.1 Завдання на лабораторну роботу

У процесі виконання лабораторної роботи для заданої предметної області необхідно виконати наступні завдання;

1) визначити призначення розподіленого програмного застосування і реалізовані бізнес процеси;

2) визначити структуру даних, що передаються між частинами розподіленого програмного застосування, відповідно до реалізованих бізнес-процесів;

3) створити коректний (well-formed) XML-документ для опису структури даних заданої предметної області;

4) створити валідну (valid) XSD-схему XML-документа;

5) створити XML-документ, валідний (valid) щодо розробленої XML схеми (XSD);

6) з використанням інтерфейсу IDE Eclipse провести валідацію XML-документа, що містить дані предметної області, на відповідність створеної XML-схемою (XSD)

7) створити таблицю стилів XSLT для відображення створеного XML-документа в браузері.

Перелік тем для вибору предметної області представлений у табл.1.

Таблиця 1 – Перелік тем для вибору предметної області

№	Тема
1.	Інформаційна система «Магазин побутової техніки»
2.	Інформаційна система «Магазин одягу»
3.	Інформаційна система «Прокат спортивного реманенту»
4.	Інформаційна система «Служба таксі»
5.	Інформаційна система «Центр сервісного обслуговування»
6.	Інформаційна система «Поліклініка»
7.	Інформаційна система «Прокат відеофільмів»
8.	Інформаційна система «Аеропорт» («ЖД вокзал»)
9.	Інформаційна система «Ресторан»
10.	Інформаційна система «Кінотеатр»
11.	Інформаційна система «Автопарк»
12.	Інформаційна система « Фітнес-Клуб»

№	Тема
13.	Інформаційна система «Хімчистка»
14.	Інформаційна система «Гуртожиток»
15.	Інформаційна система «Готель»
16.	Інформаційна система «Страхова компанія»
17.	Інформаційна система «Весільний салон»
18.	Інформаційна система «Кур'єрське агентство»
19.	Інформаційна система «Аптека»
20.	Інформаційна система «Туристична фірма»
21.	Інформаційна система «Абоненти телефонної мережі»
22.	Інформаційна система «Паспортний стіл»
23.	Інформаційна система «Склад»
24.	Інформаційна система «Розклад занять»
25.	Інформаційна система «Планувальник робочий дня»
26.	Інформаційна система «Деканат»
27.	Вільно обрана тема, погоджена з викладачем

1.4.2 Теоретичні відомості

1.4.2.1 Короткі відомості про розширювану мову розмітки (XML)

В 1986 році організацією ISO (International Organization for Standardization) був прийнятий стандарт мови SGML (ISO 8879:1986 Standard Generalized Markup Language – стандартна узагальнена мова розмітки). Він дозволяє описувати структуровані дані, організовувати й представляти інформацію, що втримується в документах, а також обмінюватися цією інформацією без втрати її «структури».

На основі SGML за завданням консорціуму W3C в 1989 році була розроблена специфікація мови HTML (Hyper Text Markup Language – мова розмітки гіпертексту), а пізніше (1996), для розширення функціональності HTML – специфікація мови XML (extensible Markup Language – розширювана мова розмітки). Опис специфікацій мови XML перебуває на сайті W3C за адресою: http://www.w3.org.

Основна мета використання XML – представити текстову інформацію за допомогою тегів, структурованих у вигляді дерева даних. Деревоподібна структура добре описує бізнес-об'єкти, конфігурацію, структури даних тощо.

Дані в такому форматі легко можуть бути як побудовані, так і розібрані для будь-якій системи з використанням будь-якої технології – для цього потрібна лише підтримка роботи з текстовими документами. На основі XML розроблені спеціалізовані стандарти обміну інформацією.

1.4.2.2 Поняття «валідних» XML-документів

XML-документ має мати строго певну структуру, що задається правилами синтаксису мови, які у свою чергу, визначаються варіантом використовуваної специфікації XML.

Коректним (well-formed) XML-документом називають документ, який відповідає специфікації XML. Якщо документ не відповідає специфікації мови, то він не може вважатися Xml-Документом.

Валидным або дійсним (valid) XML-документом називають well-formed документ, відповідний до всіх обмежень, визначених у його XML-схемі.

Крім погодженості зі специфікацією, на структуру XML-документа накладаються додаткові обмеження предметної області, що описуються за допомогою XML-схем. XML-схема визначає елементи XML-документа й атрибути, які можуть бути асоційовані з елементами. Вона також визначає структуру XML-документа, що відповідає деревоподібної (иерархичной) структурі даних предметної області. У цей час широке поширення одержали 3-ри різновиди XML-схем:

DTD-схема (Document Type Definition – визначення типу документа) дозволяє задати структуру й припустимий набір елементів в XML-документі і їх атрибутів. DTD-схема може бути оголошена як усередині XML-документа, так і поза ним. DTD має наступні недоліки: синтаксис DTD відрізняється від синтаксису XML, що утрудняє його читання й створення; DTD не підтримує опис типів даних; DTD не підтримує просторів імен XML, що дозволяють поєднувати в одному документі елементи документів різної структури.

XDR-схема (XML-Data Reduced) розроблена фірмою Microsoft для парсера власної версії специфікації мови MS XML. XML-Data – повне ім'я мови опису схем, запропонованого Microsoft консорціуму W3C (1998), а XML-Data Reduced – це частина (reduced) повної рекомендації. Microsoft також розробила й власну версію Document Content Description (DCD) for XML. XML-схеми в Microsoft Internet Explorer 5.0 і більш пізніх версіях підтримують підмножина XML-Data, що відповідає DCD. XDR-схема була оголошена як технологія перехідного періоду до широко використовуваної в цей час XSD-схемі;

XSD-схема (XML Schema Definition – визначення схеми XML), яка також часто використовує скорочену абревіатуру «XML Schema», розроблена як альтернатива DTD-схеми. XML Schema позбавлена недоліків DTD завдяки реалізації концепції «простору імен», що дозволяє задавати імена елементів без використання DTD. У мові XML Schema також реалізована розширена система типізації даних.

1.4.2.3 Структура XML-документа

Дані, що входять у документ XML, можуть бути описані в наступних розділах:

– розділ **« XML-декларація»**;

– розділ **«Пролог»**;

– розділ **«Елементи»**;

– розділ **«Атрибути»**;

– розділ **«Коментарі»**.

1.4.2.4 Декларація XML-документа

Не обов'язкова XML-декларація звичайно перебуває в першому рядку XML-документа. Вона складається з наступних елементів:

– номер версії: <?xml version="1.0"?>. Це обов'язковий аргумент, якщо вказується версія "1.1" (версія 1.0 ухвалюється за замовчуванням);

– декларація кодування: <?xml version="1.0" encoding="UTF-8"?>. Це необов'язковий параметр. Якщо він використовується, то декларація кодування повинна розташовуватися після значення атрибута version. Декларація кодування повинна містити значення, що представляє собою використовуване кодування символів (за замовчуванням – "UTF-8");

– декларація автономності: <?xml version="1.0" encoding="UTF-8" standalone="yes"?>. Декларація автономності (standalone) використовується для завдання можливості ("yes"/"no") використання зовнішнього визначення структури XML-документа за допомогою Dtd-Схем. При вказівці значення standalone="yes" Dtd-Схема розміщається в тілі Xml-Документа. Декларація автономності необов'язкова. Значення standalone за замовчуванням "yes".

1.4.2.5 Пролог XML-документа

Прологом називаються дані, розташовані після відкриваючого тегу документа або після кореневого елемента. Він включає відомості, що ставляться до документа в цілому – кодування символів, структура документа, таблиці стилів.

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

<?xml-stylesheet type="text/xsl" href="bookshop.xsl"?>

<!DOCTYPE book SYSTEM "schema.dtd">

<!--Some comments-->

У мові XML є можливість включення в документ інструкцій, які несуть певну інформацію для додатків, які будуть його обробляти. Інструкції з обробки в XML полягають у кутові лапки зі знаком питання <? уміст ?>.

1.4.2.6 Елементи XML-документа

Елементи в XML-документі відповідають за зберігання даних і є основними структурними одиницями мови XML. Елемент оформляється с допомогою парного тегу (відкриваючого <tagname> і закриваючого </tagname>), ім'я якого довільно визначається розроблювачем і полягає в кутові дужки. Синтаксис елемента з іменем тегу «tagname»:

<tagname>Уміст елемента</tagname>

Імена тегам елементів привласнюються за наступними правилами:

– імена тегів можуть містити букви, цифри й інші символи;

– імена тегів не можуть починатися із цифри або розділового знака;

– імена тегів не можуть починатися з букв «x», «m», «l», «X», «M», «L»;

– в іменах тегів не повинне бути пробілів (<tag Name> – помилка);

– не можна допускати пробілів між кутовою дужкою, що відкривається, і іменем тегу («< tagname>»,«</ tagname>», «< /tagname>» – помилка) ;

– імена тегів елементів є регістрозалежними (<tagname> і <Tagname> – різні теги);

– усі елементи повинні мати закриваючий тег.

В XML елементи можуть бути двох типів – порожні й непусті. Порожні елементи не містять у собі ніяких даних, таких як текст або інші конструкції, і можуть скорочено записуватися в такий спосіб: <Tagname /> («порожній» тег).

В XML-документі обов'язково повинен бути присутнім єдиний кореневий елемент, усі інші елементи є дочірніми стосовно єдиного кореневого елемента. При цьому теги не повинні перекриватися.

Приклад визначення єдиного кореневого елемента «bookshop» і його дочірніх елементів представлено в лістингу 1.

Лістинг 1 – Приклад визначення елементів в XML-документі

```
<?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 Атрибути XML-документа

Теги елементів XML-документа можуть мати атрибути із привласненими їм значеннями, які містяться в одинарні або подвійні лапки. Допускається використання одних лапок усередині інших. Ім'я атрибута визначається довільно й вказується через пробіл після імені тегу або попереднього атрибута. Синтаксис атрибута

<tagname attributename1="значення" attributename2='значення'>…</tagname>

Синтаксичні правила створення атрибута:

– атрибути можуть визначатися у відкриваючих або порожніх тегах, але не в закриваючих тегах;

– кількість атрибутів не обмежена;

– кілька атрибутів розділяються пробілами;

– кожне ім'я атрибута повинне бути унікально в рамках одного елемента;

– правила присвоєння імен атрибутом мають ті ж обмеження, що й імена тегів (див. вище).

Приклад визначення атрибутів представлено в лістингу 2.

Лістинг 2 – Приклад визначення атрибута «id» тегу «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 Коментарі в XML-документі

Коментарі, що не призначені для синтаксичного аналізу (наприклад, зауваження про структуру документа або редагування), можна укласти в коментарі. Коментарі починаються із групи символів «<!--» і закінчуються групою символів «-->», його синтаксис:

```
<!-- Текст коментаря -->
```

Коментарі можуть перебувати в пролозі документа, у тому числі у визначенні типу документа (DTD), після документа й у текстовому вмісті. Коментарі не можуть перебувати усередині значень атрибутів. Вони також не можуть перебувати усередині тегів.

Синтаксичний аналізатор уважає кінцем коментаря символ «>». Після цього символу він відновляє обробку XML-документа у звичайному режимі. Тому рядок «>» не може перебувати усередині коментаря. За винятком цього, у коментарях можуть використовуватися будь-які припустимі символи XML.

При створенні коментаря необхідно враховувати наступне:

– у тексті коментаря не може бути двох символів «-» підряд.

– коментар не може закінчуватися символом «-».

1.4.2.9 Текстові дані XML-документа

Завдяки підтримці набору символів Юникода XML підтримує цілий діапазон символів, у тому числі букви, цифри, розділові знаки й інші символи. Більшість керуючих символів і символи сумісності Юникода не допускаються. Увесь текст XML-документа аналізується парсером, тобто є парсируемым (PCDATA), але якщо буде потреба певні розділи можна «сховати» для перевірки й вони будуть ігноруватися парсером (т.зв. непарсируемые дані, CDATA). Розділи CDATA дають можливість повідомити засіб синтаксичного аналізу, що серед символів, що втримуються в розділі CDATA, відсутня розмітка. Це спрощує створення документів з розділами, у яких можуть з'явитися окремі символи розмітки, але насправді розмітки немає. У розділи CDATA часто поміщають уміст мовою сценаріїв, а також зразки вмісту XML і HTML.

Розділ CDATA у схемі документа використовує наступну конструкцію:

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

При виявленні початкового тегу <![CDATA[, засіб синтаксичного аналізу XML розглядає весь наступний уміст як символи, не намагаючись інтерпретувати їх як розмітку елемента або сутності. Посилання на символи в розділах CDATA не працюють. Виявивши завершальний тег]]>, засіб синтаксичного аналізу відновляє нормальний синтаксичний аналіз.

Уміст розділів CDATA може містити тільки символи, дозволені для вмісту XML; не можна екранувати в такий спосіб керуючі символи й символи сумісності. Крім того, у розділ CDATA не може входити послідовність]]>, оскільки ці символи означають завершення розділу. Це значить, що розділи CDATA не можуть бути вкладеними друг у друга. Ця послідовність також використовується в деяких сценаріях.

1.4.2.10 Сутності XML-документа

Для використання в XML-документі символів, також як і в HTML, використовуються екрановані синтаксичні конструкції, що одержали назву сутність (entity). Сутності використовуються для вистави символів, відсутніх у кодуванні XML-документа, або ж для вистави спеціальних символів.

В XML сутність – це іменовані дані, звичайно текстові, у тому числі спецсимволы. В XML існує два методи екранування спеціальних символів: посилання на сутність і посилання по номеру символу.

Посилання на сутність (entity references) вказується там, де повинна йти дана сутність. Посилання на сутність складається із символу амперсанда «&», імені сутності й крапки з коми «;».

Посилання по номеру символу (numeric character reference) також починається з амперсанда «&»і закінчується крапкою з коми «;», але спочатку йде символ «#», а далі код символу в кодуванні Юникод (у десятковій або шестнадцатеричной системі).

У табл. 1 подані приклади екранування визначених спеціальних символів. У специфікації XML тільки 5-ть символів мають визначені посилання на сутності: «&», «<», «>», «'», і «"».

Наприклад, щоб текст елемента «NURE&ST», відображався правильно необхідно використовувати сутність «&», правильний запис: «NURE&ST».

Таблиця 1. Сутності XML

Сутність	Посилання на сутність	Значення	Посилання по номеру символу
lt	**<**	**«<» – менше ніж**	<
gt	**>**	**«>» – більше чому**	>
amp	**&**	**«&» – амперсанд**	&
apos	**'**	**«'» – одиночні лапки**	'
quot	**"**	**«"» –подвійні лапки**	"
-	-	**Нерозривний пробіл**	

1.4.2.11 Простор імен XML-документа

Простір імен в XML (XML namespace) – це окремий стандарт, що має статус рекомендації, що описує іменовану групу імен тегів елементів і їх атрибутів, що служить для забезпечення їх унікальності в XML-документі.

Оскільки імена тегів елементів в XML визначаються в довільному порядку (з урахуванням правил), можливі конфлікти, коли в XML-документі використовується тег для опису двох різних типів елементів. Для того, щоб уникнути такої ситуації, був запропоновано використовувати простір імен. Приклад конфлікту імен наведено в лістингу 3.

Лістинг 3 – Конфлікт тегів <data> по типу призначення елементів

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

```
        <book>
            <title>The Emperor's</title>
            <author>Roger</author>
            <!-- Дата вступу книги на склад -->
            <data>01.08.2017</data >
        </book>
    </bookshop>
    <booklibrary>
        <book>
            <title>The Emperor's</title>
            <author>Roger</author>
            <!-- Дата виходу книги у світло -->
            <data>10.07.2017</data >
        </book>
    </booklibrary>
</books>
```

У наведеному лістингу тег елемента «date» описує дату вступу книги на склад магазину й дату виходу книги у світло.

Простору імен XML дають можливість уникнути конфлікту імен тегів елементів. Вони задаються за допомогою унікальних ідентифікаторів URI (Uniform Resource Identifier – уніфікований ідентифікатор ресурсу). Синтаксичні аналізатори XML не звертаються по цьому URI за якою-небудь додатковою інформацією, призначеної для наступної обробки документа. URI – це символьний рядок, що дозволяє тільки ідентифікувати який-небудь ресурс, у нашім випадку, елементи XML-документа. Тому URI просторів імен для різних елементів повинні бути унікальні. Слід помітити, що на відміну від URI, URL (Uniform Resource Locator – уніфікований локатор ресурсу) однозначно ідентифікує й ресурс, і його точне місцезнаходження.

Для використання простору імен використовується атрибут «xmlns» (xml namespace), значенням якого і є URI. Значення URI уважається простором імен заданим за замовчуванням. У цьому випадку всі вкладені елементи будуть належати простору імен батьківського елемента. При цьому не губиться можливість використовувати інші простори імен для дочірніх елементів. Приклад просторів імен, заданих за замовчуванням, наведено в лістингу 4.

Лістинг 4 – Приклад просторів імен, заданих за замовчуванням

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

```
    <bookshop xmlns="http://nure.ua/bookshop">
        <book>
            <title>The Emperor's</title>
            <author>Roger</author>
            <!-- Дата вступу 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 виходу книги у світло -->
            <data>10.07.2017</data>
        </book>
    </booklibrary>
<books>
```

Для спрощення роботи із просторами імен, стандарт припускає використовувати необов'язкові префікси. Префікс простору імен вказується після атрибута «xmlns» і відділяється від нього двокрапкою (лістинг 5).

Лістинг 5 – Приклад використання префіксів для просторів імен

```
<?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 вступу 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 виходу книги у світло--!>
            <Lib:data>10.07.2017</Lib:data >
        </Lib:book>
    </Lib:booklibrary>
```

```
</books>
```

Стандарт дозволяє оголосити кілька просторів імен. Приклад такого XML-документа наведено в лістингу 6, у якім для опису книги використовується однаковий тег «date», що належить різним просторам імен.

Лістинг 6 – Приклад використання декількох просторів імен

```
<?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 вступу 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 виходу книги у світло -->
          <Lib:data>10.07.2017</Lib:data >
     </Lib:book>
</books>
```

Можна також використовувати комбінацію завдання простору імен за замовчуванням і за допомогою префіксів. Змінений код попереднього прикладу представлено в лістингу 7.

Лістинг 7 – Використання декількох просторів імен за замовчуванням

```
<?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 вступу книги на склад -->
          <data>01.08.2017</data >
          <!-- Дата виходу книги у світло -->
          <Lib:data>10.07.2017</Lib:data >
     </book>
```

```
</books>
```

1.4.2.12 XML Schema Definition (XSD)

XSD-схема (XML Schema Definition – визначення схеми XML) або XML Schema – це мова опису структури XML-документа. Схема містить у собі правила граматики (або словник), що представляють собою колекцію правил, що полягають із визначень типів, а також з оголошень елементів і атрибутів. XML Schema використовується для визначення:

– елементів і їх атрибутів XML-документа;

– числа й порядку входження дочірніх елементів;

– типів даних для елементів і атрибутів;

– значень за замовчуванням для елементів і атрибутів.

Мова XML Schema заснований на синтаксисі специфікації XML і позбавлений основних недоліків мови DTD. Одним з головних переваг мови XML Schema є можливість опису типів даних. Це дозволяє:

– описати припустимий контент для документа;

– валідувати коректність даних;

– накладати обмеження на дані;

– визначати формати даних;

– конвертувати дані різних типів.

Іншою перевагою мови XML Schema є використання синтаксису XML. Завдяки цьому редагування файлів XML Schema (.xsd) здійснюється за допомогою стандартних редакторів, що підтримують XML, а для перевірки їх валидности – стандартний парсер XML.

Розглянемо приклад опису структури XML-документа з використання XML Schema. Код XML-документа подано у лістингу 8.

Лістинг 8 – Код XML-документа із прив'язкою до схеми (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 щоб використовувати посилання на файл схеми в XML-документа, у коді лістингу 8 використовуються наступні атрибути:

– bs:bookshop – кореневий елемент, для якого зазначений префікс «bs», певного для нього простору імен;

– xmlns:bs="http://nure.ua/bookshop" – атрибут xmlns визначає простір імен «http://nure.ua/bookshop» кореневого елемента «bookshop» XML-документа (із префіксом «bs»);

xmlns:xsi="http://www.w3.org/2001/Xmlschema-instance" – атрибут xmlns визначає простір імен (за замовчуванням) для схеми XML Schema із префіксом «xsi»; Цей простір імен містить елементи й атрибути Xmlschema, які можна включати в документ XML. За загальною згодою префікс xsi використовується для цього простору імен і додається на початку імен усіх елементів і атрибутів, що належать простору імен, відділяючись від них двокрапкою;

– xsi:schemalocation="http://nure.ua/bookshop Bookshop.xsd" – атрибут schemalocation із префіксом «xsi»; дозволяє вказати простанство імен XML-документа, а через пробіл шлях до файлу й ім'я файлу XML Schema.

У лістингу 9 наведений код XML Schema, що зберігається у файлі «Bookshop.xsd».

Лістинг 9 – Код XML Schema (файл Bookshop.xsd)

```
<!-- == Xml-Декларація====================================== -->
<?xml version="1.0" encoding="UTF-8"?>
<!-- == Корнєвій елемент schema ============================ -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/Xmlschema"
            targetnamespace="http://nure.ua/bookshop"
            xmlns:bs="http://nure.ua/bookshop"
            elementformdefault="qualified">
<!-- ==Розділ опису елементів============================== -->
<!-- ==Комплексний елемент bookshop ======================== -->
<xsd:element name="bookshop">
      <xsd:complextype>
            <xsd:sequence>
            <xsd:element name="book" type="bs:Book" maxoccurs="unbounded"/>
            </xsd:sequence>
      </xsd:complextype>
</xsd:element>
<!-- ==Комплексний елемент 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>
<!-- == Опису атрибута id ============================== -->
      <xsd:complextype name="Entity" abstract="true">
            <xsd:attribute name="id" type="bs:Entityid" use="required" />
      </xsd:complextype>
<!-- ==Простий елемент Price ============================== -->
      <xsd:simpletype name="Price">
            <xsd:restriction base="xsd:double">
                  <xsd:mininclusive value="0" />
            </xsd:restriction>
      </xsd:simpletype>
<!-- ==Простий елемент 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>
<!-- ==Простий елемент 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>
```

Елементи XSD-схеми називають компонентами (components), щоб відрізнити їх від елементів описуваного документа XML. Кореневий компонент схеми має ім'я «schema». У специфікації XSD уміст компонента «schema» визначається наступними розділами:

– «type definitions» – опис застосовуваних типи даних (прості, комплексні, складові);

– «attribute declarations» – опис глобальних атрибутів;

– «element declarations» – опис елементів, що входять до складу Xml-Документа;

– «attribute group definitions» – опис використовуваних груп атрибутів;

– «model group definitions» – опис використовуваних груп;

– notation declarations – опис додаткових елементів схеми (шаблони вистави значень);

– annotations – анотації.

Таким чином, спочатку визначаються застосовувані типи даних, як прості, комплексні або складові. Потім випливає оголошення глобальних атрибутів, які ставляться до всіх елементів відразу. Після цього випливає розділ оголошення елементів, що входять до складу XML-документа. Слідом за ними оголошуються використовувані групи атрибутів. Потім описуються додаткові елементи схеми, такі як шаблони значень елементів, або ж моделі вистави. І в самому кінці розміщаються так звані анотації.

У якості окремих компонентів схеми застосовуються 13-ть типів компонентів, що розбиваються звичайно на три групи:

– до першої групи ставляться визначення простих типів, визначення складених типів, оголошення атрибутів і оголошення елементів;

– до другої групи ставляться визначення груп атрибутів, визначення унікальності елементів, визначення моделей вистави й так званих нотацій, тобто елементів, що розшифровують призначення того або іншого елемента;

– у третю групу входять анотації, групи моделей вистави інформації, додаткові дані про моделі й оголошення шаблонів вистави вмісту елементів.

Кореневим елементом у схемі XML є елемент «schema», який містить усі інші елементи в документі схеми. У рамках кореневого елемента атрибутом «xmlns» визначається простір імен Xmlschema, яке містить елементи й атрибути Xsd-Схеми.

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

Усі елементи XSD починаються із префікса «xsd:», який вказується для простору імен XSD, оголошеного в кореневому елементі екземпляра схеми.

Наступний атрибут «targetnamespace» тегу «schema»:

```
targetnamespace="http://nure.ua/bookshop"
```

визначає цільовий простір імен Xml-Документа, для якого створюється Xsd-Схема.

Атрибут «xmlns» тегу «schema»:

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

посилається на простір імен із префіксом «:bs» Xml-Документа, для якого створюється XSD-схема.

Атрибут «elementformdefault» тегу «schema»:

```
elementformdefault="qualified"
```

задає значення, повинні чи елементи XML-документа, для якого визначена XSD-схема, належати простору імен, зазначеному в атрибуті targetnamespace. Він може ухвалювати 2-ва значення:

«qualified» – усі елементи повинні належати цільовому простору імен

«unqualified» – усі елементи схеми не належать цільовому простору імен(значення за замовчуванням). Для кожного елемента потрібно явно вказати приналежність простору імен, якщо необхідно.

1.4.2.13 Елементи XML-схем

Опис структури XML-документа в цілому зводиться до опису елементів і їх атрибутів. При цьому повинна враховуватися ієрархія входжень елементів друг у друга. При описі схем XML-документів спочатку визначаються найпростіші елементи, які не містять дочірніх субелементів. Тобто перед описом якогось батьківського елемента спочатку описуються всі його дочірні субелементи. Дана концепція природно запозичена з високоструктурованих мов програмування, у яких не можна використовувати змінні, не оголосивши їх попередньо.

В XSD схемі (див. лістинг 9) перший рядок містить XML-декларацію. Після декларації завжди обов'язково визначається кореневий елемент «schema». Далі визначається тип елемента за допомогою тэга <Eiementtype> з обов'язковим параметром name. У якості строкового значення цього параметра використовується найменування єдиного кореневого тегу XML-документа для якого створюється схема. Теги для опису елементів верхнього рівня XML-схем представлені в табл. 2.

Таблиця 2. Теги для опису елементів верхнього рівня XML-схем

Елемент	Опис
<xsd:annotation>	**Визначає замітку.**
<xsd:attribute>	**Повідомляє атрибут.**
<xsd:attribute>	**Групує набір оголошень атрибутів таким чином, що їх можна включити в якості групи у визначення складних типів.**
<xsd:complextype>	**Повідомляє складний тип, що визначає набір атрибутів і вміст елемента.**
<xsd:element>	**Повідомляє елемент, який може включати наступні атрибути: name — обов'язковий атрибут, визначає ім'я елемента; type — указує тип елемента; ref — посилається на визначення елемента, що перебуває в іншім місці; minoccurs і maxoccurs — кількість повторень цього елемента (за замовчуванням ухвалює значення «1»). Щоб указати, що кількість елементів не обмежена, в атрибуті maxoccurs необхідно задати unbounded**
<xsd:group>	**Групує набір оголошень елементів таким чином, що їх можна включити в якості групи у визначення складних типів**
<xsd:import>	**Визначає простір імен, на компоненти схеми якого посилається утримуюча схема.**
<xsd:include>	**Включає зазначений документ схеми в цільовий простір імен утримуючої схеми.**
<xsd:notation>	**Містить визначення нотації, що описує формат не-XML даних в Xml-Документі. Визначення нотації схеми XML – це видозміна визначень XML 1.0 NOTATION.**
<xsd:redefine>	**Дозволяє перевизначити в поточній схемі прості й складні типи, групи й групи атрибутів, отримані із зовнішніх файлів схем.**
<xsd:simpletype>	**Оголошення елементів простих типів, які не мають атрибутів і дочірніх елементів.**

Структура компонента «element» XSD-схеми представлено в лістингу 10.

Лістинг 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>
```

Процес створення XSD-схеми містить у собі два кроки – визначення й оголошення типів елементів або типів атрибутів. Елементи й атрибути XML-документа оголошуються елементами схеми «xsd:element» і «xsd:attribute». Структура ж XML-документа визначається елементами схеми «xsd:simpletype» і «xsd:complextype».

Основне оголошення елемента складається з імені й типу даних:

```
<xsd:element name="ім'я_елемента" type="xsd:тип_даних"/>
```

В XSD- схемах теги (дескриптори), використовувані в Xml-Документах, розділяються на дві категорії типів складні й прості. Елементи складних типів можуть містити інші елементи, а також мають певні атрибути; елементи простих типів такими можливостями не мають.

1.4.2.14 Прості (simple) типи XML-схем

Елементи, які не мають атрибутів і дочірніх елементів, називаються простими й повинні мати простий тип даних.

Теги для опису простих типів елементів XML-схем представлені в табл. 3.

Таблиця 3. Теги для опису простих типів XSD-схем

Елемент	Опис
<xsd:annotation>	**Визначає замітку.**
<xsd:appinfo>	**Задає відомості, використовувані додатками в елементі annotation.**
<xsd:documentation>	**Задає відомості, які читають або використовують користувачі в елементі annotation.**
<xsd:element>	**Повідомляє елемент.**
<xsd:list>	**Визначає колекцію з одного визначення simpletype.**
<xsd:restriction> (simpletype)	**Задає обмеження на визначення simpletype.**
<xsd:union>	**Визначає колекцію з декількох визначень simpletype.**

Синтаксис простого елемента:

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

де «xxx» – ім'я елемента, а «yyy» – його тип.

Є дві головні категорії простих типів: вбудовані типи й певні користувачем прості типи.

Мова XSD має велика кількість вбудованих простих типів даних. Вбудовані типи містять у собі примітивні типи й похідні. Примітивні типи даних не отримані з інших типів даних. Наприклад, числа із плаваючої коми – математичне поняття, яке не отримане з інших типів даних. Похідні типи даних визначені в термінах існуючих типів даних. Наприклад, ціле число – окремий випадок, отриманий з десяткового типу даних.

У табл. 4 представлений список примітивних типів даних Xsd-Схеми, які можуть бути застосовані до типу даних і для опису типу даних.

Таблиця 4 – Примітивні типи даних XSD-схем

Тип даних	Обмеження	Опис
string	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє символьний рядок.**
Boolean	**pattern, whitespace**	**Представляє логічне значення, яке може бути true або false.**
decimal	**enumeration, pattern, totaldigits, fractiondigits, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє довільне число.**
float	**pattern, enumeration, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє 32-бітове число із плаваючої коми одиночної точності.**
double	**pattern, enumeration, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє 64-бітове число із плаваючої коми подвійної точності.**

Тип даних	Обмеження	Опис
duration	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє тривалість часу. Шаблон для duration наступний - Pnynmndtnhnmns, де ny представляє число років; nm - місяців; nd - днів; T - роздільник дати й часу; nh - число годин; nm - хвилин; ns - секунд.**
datetime	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє конкретний час. Шаблон для datetime наступний - Ccyy-mm-ddthh:mm:ss, де CC представляє сторіччя; YY - рік; MM - місяць; DD - день; T - роздільник дати й часу; hh - число годин; mm - хвилин; ss - секунд. При необхідності можна вказувати частки секунди. Наприклад, соті частки в шаблону: ss.ss**
time	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє конкретний час дня. Шаблон для time наступний -hh:mm:ss.sss (часткова частина секунд необов'язкова).**
date	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє календарну дату. Шаблон для date такий - CCYY-MM-DD (тут необов'язкова частина, що представляє час).**
gyearmonth	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє конкретний місяць конкретного року (CCYY-MM).**
gyear	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє календарний рік (CCYY).**
gmonthday	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє конкретний день конкретного місяця (--MM-DD).**
gday	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє календарний день (---DD).**

Тип даних	Обмеження	Опис
gmonth	**enumeration, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, whitespace**	**Представляє календарний місяць (--ММ--).**
hexbinary	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє довільну шестнадцатерично-закодовану двійкову інформацію. Hexbinary - набір двійкових октетів фіксованої довжини, що полягає із чотирьох пар шестнадцатеоисных символів. Наприклад, 0-9a-fa-f.**
base64Binary	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє довільну Base 64-закодовану двійкову інформацію. Base64Binary - набір двійкових октетів фіксованої довжини.**
anyuri	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє URI як визначено в RFC 2396. Значення anyuri може бути абсолютно або відносно, і може мати необов'язковий ідентифікатор фрагмента.**
Qname	**length, enumeration, pattern, maxlength, minlength, whitespace**	**Представляє складене ім'я. Ім'я складене із префікса й локальної назви, відділеного двокрапкою. І префікс і локальні назви повинні бути NCNAME. Префікс повинен бути пов'язаний з namespace URI посиланням, використовуючи оголошення простору імені.**
NOTATION	**length, enumeration, pattern, maxlength, minlength, whitespace**	**Представляє тип атрибута СИСТЕМИ ПОЗНАЧЕНЬ. Набір QNAMES.**

У табл.5 наведена XSD-схема, що містить визначення простих типів.

Таблиця 5 – Використання примітивних типів даних

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"/>`

У табл.6 представлений список похідних типів даних XSD-схеми й обмежники, які можуть бути застосовані до типу даних і в описі типу даних.

Таблиця 6 – Похідні типи даних XSD-схеми

Тип даних	Обмеження	Опис
normalizedstring	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє нормалізовані рядки. Цей тип даних отриманий з string.**
token	**enumeration, pattern, length, minlength, maxlength, whitespace**	**Представляє маркіровані рядки. Цей тип даних отриманий з normalizedstring.**
language	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє ідентифікатори природньої мови (певний RFC 1766). Цей тип даних отриманий з token**
IDREFS	**length, maxlength, minlength, enumeration, whitespace**	**Представляє тип атрибута IDREFS. Містить набір значень типу IDREF.**
ENTITIES	**length, maxlength, minlength, enumeration, whitespace**	**Представляє тип атрибута ENTITIES. Містить набір значень типу ENTITY.**
NMTOKEN	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє тип атрибута NMTOKEN. NMTOKEN - набір символів імен (символи, цифри й інші символи) у будь-якій комбінації. У відмінність від Name і NCNAME, NMTOKEN не має ніяких обмежень на перший символ. Цей тип даних отриманий з token.**
NMTOKENS	**length, maxlength, minlength, enumeration, whitespace**	**Представляє тип атрибута NMTOKENS. Містить набір значень типу NMTOKEN.**
Name	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Представляє імена в XML. Name - лексема(маркер), яка починається із символу, символу підкреслення або двокрапки й триває символами імен (символи, цифри, і інші символи). Цей тип даних отриманий з token.**
Ncname	**length, pattern, maxlength, minlength, enumeration, whitespace**	**Цей тип даних - той же, що й Name, але не може починатися із двокрапки. Цей тип даних отриманий з Name.**
ID	**length, enumeration, pattern, maxlength, minlength, whitespace**	**Представляє тип атрибута ID, певний в XML 1.0 Рекомендації. ІДЕНТИФІКАТОР не повинен мати двокрапки (Ncname) і повинен бути унікальний у межах XML документа. Цей тип даних отриманий з NCNAME.**
IDREF	**length, enumeration, pattern, maxlength,**	**Представляє посилання до елемента, що має атрибут ID, який точно відповідає**

Тип даних	Обмеження	Опис
	minlength, whitespace	встановленому ІДЕНТИФІКАТОРУ. IDREF повинен бути NCNAME і повинен бути значенням елемента або атрибута типу ID у межах XML документа. Цей тип даних отриманий з NCNAME.
ENTITY	length, enumeration, pattern, maxlength, minlength, whitespace	Представляє тип атрибута ENTITY. Це - посилання до неаналізованого об'єкта з іменем, яке точно відповідає встановленому імені. ENTITY повинен бути NCNAME і повинен бути оголошений у схемі як неаналізоване ім'я об'єкта. Цей тип даних отриманий з NCNAME.
integer	enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	Представляє послідовність десяткових цифр із необов'язковим знаком (+ або -). Цей тип даних отриманий з decimal.
nonpositiveinteger	enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	Представляє ціле число, менше або рівне нулю. Nonpositiveinteger складається з негативного знака (-) і послідовності десяткових цифр. Цей тип даних отриманий із цілого числа.
negativeinteger	enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	Представляє ціле число, менше нуля. Цей тип даних отриманий з nonpositiveinteger.
long	enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	Представляє ціле число з мінімальним значенням -9223372036854775808 і максимумом 9223372036854775807. Цей тип даних отриманий із цілого числа.
int	enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	Представляє ціле число з мінімальним значенням -2147483648 і максимумом 2147483647. Цей тип даних отриманий з long.
short	enumeration, fractiondigits, pattern,	Представляє ціле число з мінімальним значенням -32768 і максимумом 32767. Цей

Тип даних	Обмеження	Опис
	mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	**тип даних отриманий з int.**
byte	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число з мінімальним значенням -128 і максимумом 127. Цей тип даних отриманий з short.**
nonnegativeinteger	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число, більше рівне нулю. Цей тип даних отриманий із цілого числа.**
unsignedlong	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число з мінімумом нуль і максимумом 18446744073709551615. Цей тип даних отриманий з nonnegativeinteger.**
unsignedint	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число з мінімумом нуль і максимумом 4294967295. Цей тип даних отриманий з unsignedlong.**
unsignedshort	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число з мінімумом нуль і максимумом 65535. Цей тип даних отриманий з unsignedint.**
unsignedbyte	**enumeration, fractiondigits, pattern, mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace**	**Представляє ціле число з мінімумом нуля й максимуму 255. Цей тип даних отриманий з unsignedshort.**
positiveinteger	**enumeration, fractiondigits, pattern,**	**Представляє ціле число, яке є більшим чому нуль. Цей тип даних отриманий з**

Тип даних	Обмеження	Опис
	mininclusive, minexclusive, maxinclusive, maxexclusive, totaldigits, whitespace	**nonnegativeinteger.**

Отримані із вбудованих типів, застосуванням до них іменованих обмежень, називаними аспектами (facets). Аспекти обмежують припустимі значення простих типів. Синтаксис застосування аспектів обмеження наступний:

```
<xsd:restriction base="тип_даних">
     <xsd:ім'я_аспекту value="значення_аспекту"/>
</xsd:restriction>
```

Для вказівки основного типу використовується елемент «restriction». Його атрибут «base» указує на основний тип. В елемент «restriction» можна включити ряд обмежень на значення типу (табл. 7).

Таблиця 7 –Типи дозволених обмежень, залежно від типу

Обмеження	Опис	Приклад
xsd:enumeration	**Визначає список припустимих значень**	**<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	**Визначає максимальне число цифр після коми. Повинне бути більше або рівно 0.**	
xsd:maxexclusive	**Визначає верхню границю для цифрових величин (значення повинне бути менше зазначеної величини)**	
xsd:minexclusive	**Визначає мінімальне значення для цифрових значень (значення повинне бути більше зазначеної величини)**	
xsd:maxinclusive	**Визначає верхню границю для цифрових величин (значення повинне бути менше або рівно зазначеної величини)**	**<xsd:element name="age"> <xsd:simpletype> <xsd:restriction base="xsd:integer">**

Обмеження	Опис	Приклад
xsd:mininclusive	**Визначає найменше дозволене значення для цифрових величин (значення повинне бути більше або дорівнює зазначеній величині**	**<xsd:mininclusive value="0"/> <xsd:maxinclusive value="120"/> </xsd:restriction> </xsd:simpletype> </xsd:element>**
xsd:length	**Визначає точне количесто символів або елементів у списку. Повинне бути більше або рівно 0.**	**<xsd:element name="password"> <xsd:simpletype> <xsd:restriction base="xsd:string"> <xsd:length value="8"/> </xsd:restriction> </xsd:simpletype> </xsd:element>**
xsd:maxlength	**Визначає максимальне число символів у значенні або елементів у вписке. Повинне бути більше або рівно 0.**	**xsd:element name="password"> <xsd:simpletype> <xsd:restriction base="xsd:string"> <xsd:minlength value="5"/> <xsd:maxlength value="8"/>**
xsd:minlength	**Визначає мінімальне количиство символів або елементів у списку. Повинне бути більше або рівно 0.**	**</xsd:restriction> </xsd:simpletype> </xsd:element>**
xsd:pattern	**Визначає регулярне вираження, що визначає дозволені значення**	
xsd:totaldigits	**Визначає точна кількість цифр у цифровій величині. Повинне бути більше 0.**	
xsd:whitespace	**Визначає як обробляються символи роздільники (переклади рядка, табуляції, пробіли, повернення 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"	**Ніяка нормалізація не виконується.**
	value="replace"	**Усі #x9 (tab), #xa (line feed) and #xd (carriage return) заміняються на #x20 (пробіл).**
	value="collapse"	**Після replace-обробки всі внутрішні ланцюжки #x20 руйнуються до одного пробілу, а навколишні пробіли віддаляються.**

Аспекти можуть бути зазначені тільки одного разу у визначенні типу, крім enumeration і pattern - вони можуть мати багаторазові входження й групуються.

У мові XSD реалізована концепція іменованих типів. Наприклад, при створенні визначення, можна привласнити цьому визначенню ім'я, щоб повторно використовувати його в XSD-схемі.

Розглянемо приклад створення елемента простого типу (simpletype) з іменем «txt15pre». У результаті буде створено іменоване обмеження. Після цього з'являється можливість застосовувати це обмеження й до інших елементів у схемі. Це корисно, коли у визначенні застосовуються аспекти обмеження типу даних, щоб не повторювати їхнім щораз в інших визначеннях. Наприклад, елемент «simpletype» може бути пов'язаний з елементом «Прізвище» і атрибутом «Телефон» для оголошення змісту цих елемента й значення атрибута як строкових даних:

```
<xsd:simpletype name="txt15pre">
     <xsd:restriction base="xsd:string">
          <xsd:maxlength value="15"/>
          <xsd:whitespace value="preserve"/>
     </xsd:restriction>
</xsd:simpletype>
<xsd:element name="Прізвище" type="txt15pre"/>
<xsd:attribute name="Телефон" type="txt15pre" use="required"/>
```

Звернули увагу на ключове слово в оголошенні атрибута? Як і в попередніх схемах, значення атрибута use="required" усі так само означає обов'язковість використання оголошеного атрибута. Іншими визначеними значеннями атрибута «use» елемента схеми «xsd:attribute» можуть бути ключові слова «optional» і «prohibited». Перше з них означає необов'язковість використання, а друге – забороняє використання оголошеного атрибута. Така необхідність виникає у випадку локального оголошення раніше певної групи атрибутів елементом схеми «xsd:attributegroup», наприклад:

```
<xsd:attributegroup name="Зв'язок">
       <xsd:attribute name="Телефон" type="txt15pre"/>
       <xsd:attribute name="Факс" type="txt15pre"/>
</xsd:attributegroup>
```

Далі в контексті визначення елемента складного типу робиться обмеження на застосування атрибутів цієї групи:

```
<xsd:complextype name="Клієнт">
     <xsd:complexcontent>
          <xsd:restriction base="xsd:Зв'язок">
               <xsd:attribute name="Телефон" use="required"/>
               <xsd:attribute name="Факс" use="prohibited"/>
          </xsd:restriction>
     </xsd:complexcontent>
</xsd:complextype
```

1.4.2.15 Комплексні (complex) типи елементів XML-схем

Комплексні (складні) типи елементів XML-схем – це елементи, що містять вкладені елементи або атрибути.

Складні елементи створюються за допомогою елемента «complextype». Так само, як і в простому типі атрибут «name» задає ім'я типу. Для вказівки, що елементи усередині описуваного складного типу повинні розташовуватися в певній послідовності, використовуються елементи «sequence», «all», «choice». Він може містити елементи «element», що визначають зміст складного типу.

Якщо тип може містити не тільки елементи, але й текстову інформацію, необхідно задати значення атрибута «mixed="true"». Крім елементів, тип може містити атрибути, які створюються елементом «attribute». Атрибути елемента «attribute»: «name» – ім'я атрибута, «type» – тип значення атрибута. Для вказівки, чи зобов'язано використовуватися атрибут, потрібно використовувати атрибут «use», який ухвалює значення «required», «optional», «prohibited». Для установки значення за замовчуванням використовується атрибут «default», а для фіксованого значення – атрибут «fixed».

У табл. 8 наведені елементи, що створюють визначення комплексних (складних) типів.

Таблиця 8. Визначення складних типів Xsd-Схем

Елемент	Опис
<xsd:all>	**Дозволяє елементам групи з'являтися (або не з'являтися) в утримуючому елементі в будь-якому порядку.**
<xsd:annotation>	**Визначає замітку.**
<xsd:any>	**Дозволяє будь-якому елементу із зазначених просторів імен з'являтися в утримуючому їхньому елементі sequence або choice.**
<xsd:anyattributc>	**Дозволяє будь-якому атрибуту із зазначених просторів імен з'являтися в утримуючому їхньому елементі complextype або attributegroup.**
<xsd:appinfo>	**Задає відомості, використовувані додатками в елементі annotation.**
<xsd:attributc>	**Повідомляє атрибут.**
<xsd:attributc>	**Групує набір оголошень атрибутів таким чином, що їх можна включити в якості групи у визначення складних типів.**
<xsd:choice>	**Дозволяє бути присутнім в елементі-контейнері одному й тільки одному елементу обраної групи.**
<xsd:complexcontent>	**Містить розширення або обмеження для складного типу, що зберігає змішане вміст або тільки елементи.**

Елемент	Опис
<xsd:documentation>	**Задає відомості, які читають або використовують користувачі в елементі annotation.**
<xsd:element>	**Повідомляє елемент.**
<xsd:extension> (simplecontent)	**Містить розширення simplecontent. Виконується розширення простого або складного типу, що містить простий уміст, шляхом додавання зазначених атрибутів, груп атрибутів, або атрибута anyattribute.**
<xsd:extension> (complexcontent)	**Містить розширення для complexcontent**
<xsd:group>	**Групує набір оголошень елементів таким чином, що їх можна включити в якості групи у визначення складних типів**
<xsd:restriction> (simplecontent)	**Задає обмеження на визначення simplecontent.**
<xsd:restriction> (complexcontent)	**Задає обмеження на визначення complexcontent.**

Модель змісту елемента складного типу – формальний опис структури й припустимого змісту елемента, який використовується для перевірки правильності XML-документа. Модель змісту може обмежувати документ до деякого набору елементних типів і атрибутів, описувати й підтримувати зв'язки між цими різними компонентами й унікально позначати окремі елементи. Вільне використання моделі змісту дозволяє розроблювачам змінювати структурну інформацію.

Перелік оголошень дочірніх елементів приводиться в структурі, що групує XSD-елементи «choice», «sequence», і «all».

Елемент «xsd:choice» дозволяє тільки одному з елементів, що втримуються в групі бути присутнім у складі елемента XML-документа. Елемент «xsd:sequence» вимагає появи елементів групи в точно встановленій послідовності в складі елемента XML-документа. Елемент «xsd:all» дозволяє субэлементам у групі бути (або не бути) у будь-якому порядку в складі елемента XML-документа.

Елемент «xsd:group» використовується для чіткого визначення групи й для посилання до іменованої групи. Використання моделі групи дозволяє визначити набір елементів, які можуть бути повторені в XML-документі. Це корисно для формування визначення комплексного типу. Іменовану модель групи можна далі визначити, використовуючи «xsd:sequence», «xsd:choice» або «xsd:all» у дочірніх елементах. Іменовані групи повинні визначатися в корені схеми. При необхідності багаторазового використання переліку елементів,

певного в групі досить дати посилання на іменовану групу «xsd:group ref="ім'я_групи"»

Визначення складних типів створюються з використанням елемента «complextype», його атрибутів і будь-яких припустимих аспектів (обмежень). Звичайно, складні типи містять набір оголошень елементів, оголошень атрибутів і елементних посилань.

```
<xsd:element name="ім'я_елемента" type="xsd:тип_даних">
     <xsd:complextype>
          <xsd:sequence>
               <xsd:element name="ім'я_елемента" type="xsd:тип_даних"/>
          </xsd:sequence>
     <xsd:attribute name="ім'я_атрибута" type="xsd:тип_даних"/>
     </xsd:complextype>
</xsd:element>
```

У табл. 9 наведений приклад визначення XSD-схеми для складного типу.

Таблиця 9 – Приклад визначення складних типів 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> <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 Атрибути XML-схем

У табл. 10 наведені елементи, що визначають атрибути в схемах.

Таблиця 10 – Елементи, що визначають атрибути XML-схем

Елемент	Опис
<xsd:anyattribute>	Дозволяє будь-якому елементу із зазначених просторів імен з'являтися в утримуючому їхньому елементі complextype або attributegroup.
<xsd:attribute>	Повідомляє атрибут.
<xsd:attribute>	Групує набір оголошень атрибутів таким чином, що їх можна включити в якості групи у визначення складних типів.

Усі атрибути декларуються, як прості типи. Прості елементи не можуть мати атрибути. Якщо в елемента є атрибути, то він ставиться до комплексних або складеним типам. Але сам по собі атрибут завжди декларується, як простий тип.

Атрибут декларується в такий спосіб:

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

де «xxx» – ім'я атрибута, а «yyy» – тип даних атрибута.

Типи даних, використовувані в атрибутах, збігаються з деклараціями простих типів і розглянуті вище.

Приклад опису атрибута наведений у табл. 11.

Таблиця 11 –Приклад опису атрибутів

Xml-Документ	`<lastname lang="EN">Smith</lastname>`
Xsd-Схема	`<xsd:attribute name="lang" type="xsd:string"/>`

Атрибути можуть мати значення за замовчуванням або фіксовані значення. Значення за замовчуванням привласнюється атрибуту автоматично, якщо не визначене ніякого іншого значення.

Приклад оголошення значення за замовчуванням ("EN"):

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

Фіксоване значення також привласнюється атрибуту автоматично, але при цьому ви не можете визначити ніякого іншого значення.

Приклад оголошення фіксованого значення ("EN"):

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

За замовчуванням атрибути є необов'язковими для використання. Щоб декларувати обов'язковий атрибут, слід скористатися атрибутом «use»:

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

Якщо XML елемент або атрибут має визначення типу даних, то це накладає обмеження по контенту цього елемента або атрибута. Наприклад, якщо Xml-Елемент має тип «xsd:date» і містить рядок, наприклад, «Hello World», те цей елемент не пройде валидацию або перевірку на коректність даних.

1.4.3 Порядок виконання роботи

Завдання лабораторної роботи необхідно виконувати поетапно. Зразкова послідовність етапів роботи:

1. Вибрати тему, що визначає предметну область (табл.1).
2. Визначити призначення розподіленого програмного додатка. Скласти перелік реалізованих бізнес процесів.
3. Визначити структуру й типи даних, переданих між частинами розподіленого додатка, відповідно до реалізованих бізнесів-процесів.
4. З використанням IDE Eclipse створити проект, що включає Xml-Документ. Провести валідацію XML-документа.
5. З використанням IDE Eclipse створити XSD-схему XML-документа. Провести валидацию Xsd-Схеми.
6. З використанням IDE Eclipse згенерувати XML-документ на основі XSD-схеми. Зробити порівняння згенерованого й створеного в пп.4 XML-документів. При необхідності зробити коректування XSD-схеми. і повторити процес порівняння.
7. Заповнити згенерований і валідний XML-документ даними.
8. Оформити звіт за роботою.

1.5 Зміст звіту

Звіт має містити:

- ціль роботи;
- постановку задачі;
- опис предметної області й перелік реалізованих бізнес-процесів;
- код файлу з XSD-схемою;
- код згенерованого на основі XSD-схеми XML-файлу, із внесеними даними;
- результати валідації XSD-схеми й XML-файлу;
- висновки за роботою.

1.6 Критерії оцінки для оформленого звіту (без теорії)

Створено коректний (well-formed) і валідний (valid) XML-документ;

Створена коректна (well-formed) і валідна (valid) XSD-схема XML-документа;

У XSD-схемі при описі елементів XML-документа:

- визначено простір імен та його префікси;
- описані всі елементи XML-документа, із зазначенням типу, обмежень (minOccurs, maxOccurs, minInclusive, minExclusive, maxInclusive, maxExclusive, length, maxLength, minLength, fractionDigits, totalDigits), регулярних виразів (<xsd: pattern>), а також з використанням і без використання <xsd: restriction>;
- при описі компонентів складних типів використані всі 3-ри інструкції: <xsd: all>, <xsd: choice>, <xsd: sequence>;
- має бути описаний компонент (<xsd: element>), що містить визначений список (<xsd: enumeration>) можливих прийнятих значень;
- описаний обов'язковий атрибут, його тип і обмеження на значення;
- описаний необов'язковий атрибут, його тип і обмеження на значення;

4) створена таблиця стилів XSLT, яка дозволяє відобразити XML-документ в браузері. При цьому використовуються різні формати даних (шрифт: розмір шрифту, накреслення; таблиці, параграфи, списки; реалізована фільтрація даних).

1.7 Контрольні питання й завдання

1. Що можуть містити сутності, з яких полягає XML-документ?
2. Для чого в XML-документах використовуються теги?
3. Що йде перед кореневим елементом XML-документа?
4. Що таке початковий і кінцевий теги?
5. Як можна записати порожній елемент?
6. Що може бути іменем елемента або атрибута в XML-документі?
7. Який XML-документ є коректним.
8. Який XML-документ є валидным.
9. Як в XML-документі позначаються коментарі?
10. Припустимі чи в XML-документі не закриті теги?
11. Що таке теги, що перекриваються, і чи припустимі вони в XML-документах?
12. Що таке спеціальні символи в XML-документах?
13. Як можна екранувати спеціальні символи в XML-документах?
14. Що таке посилання на сутність?
15. Скільки визначених сутностей є в мові XML?
16. Які вбудовані типи мови опису XML-схем використовуються в цей час?
17. Що розуміється під XML Schema?
18. Для чого використовується XML Schema?
19. Які переваги XML Schema перед DTD?
20. Як підключити XML Schema до XML-документу?
21. Який синтаксис оголошення простого елемента в XML Schema?
22. Які типи даних існують в XML Schema?
23. Описати простий тип, який ґрунтується на positiveinteger. Припустимі значення для даного типу лежать у діапазоні [771, 116].
24. Описати простий тип, для якого припустимим значенням буде рядок з 5 букв у нижньому регістрі.
25. Описати простий тип, для якого припустимим буде одне з перерахованих значення: Caption1, Caption2, Caption3.
26. Приведіть XSD-схему для наступного елемента

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

де id і name – обов'язкові атрибути, pages – необов'язковий.

27. Приведіть XSD-схему для наступного елемента

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

28. Які model group виділяють для опису вкладених елементів. Чим вони відрізняються (<sequence>, <all>, <choice>).

29. Приведіть XSD-схему для наступного елемента

```
<person id=”4546464”>
    <name>Ivan</name>
    <surname>Ivanov</surname>
    <birthday>01.01.1992</birthday>
    <phone>0953682547</phone>
    <phone>0679871236</phone>
</person>
```

30. Яким образом в XSD реалізований механізм спадкування? Приведіть приклад.

Навчальне видання / Електронне навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторної роботи № 1
з дисципліни
«INTERNET ТЕХНОЛОГІЇ»

Укладачі: МІЩЕРЯКОВ Юрій Валентинович
КОВАЛЕНКО Андрій Іванович
РЕШЕТНИК Віктор Михайлович

Відповідальний випускаючий: Гребенник І.В.