Урок 3

План заняття:

1. Простори імен XML
2. Поняття XML схем. Відмінності схем від DTD. Огляд існуючих XML схем
3. Схеми XSD (стандарт W3C). Опис заголовка, елементів та атрибутів. Вбудовані типи даних XSD-схем
4. Прості та комплексні типи даних
5. Звуження типів
6. Списки та об'єднання
7. Визначення складних типів. Моделі грипи. Розширення та звуження складних типів
8. Посилання на елементи
9. Анотації
10. Підключення XML схеми до документа
11. Приклади створення XSD схем
12. Домашнє завдання

1. Простори імен XML

Сьогоднішнє наше обговорення ми розпочнемо з розмови про простори імен. Гадаю, не зайвим буде нагадати, що в різних мовах розміток, тобто в різних реалізаціях XML документів можуть зустрічатись одні і ті ж імена тегів та їх атрибутів, які при цьому мають різне смислове навантаження. Прикладом такої ситуації може слугувати факт розробки одного документа різними розробниками, які можуть використати елементи або атрибути з однаковими іменами, але які використовуються для різних цілей. Для рішення цих проблем в XML використовують **простори імен (namespace)**.

Кожному простору імен присвоюється **унікальний ідентифікатор ресурса (URI)**. Щоб розрізнити елементи одного простору імен від іншого, цей унікальний ідентифікатор вказується в якості префікса відповідних імен елементів та атрибутів та відділяється від самого імені двокрапкою. Отже, всі імена елементів та атрибутів, префікси яких зв'язані з одним і тим же ідентифікатором, утворюють один простір імен.

Префікс та ідентифікатор простору імен визначаються атрибутом **xmlns** наступним чином:

```
<префікс:елемент xmlns:префікс ="унікальний ідентифікатор">
</префікс:елемент>
```

Але в жодному разі ми не можемо написати щось на зразок:

```
<префікс1:префікс2:елемент xmlns:префікс ="унікальний ідентифікатор">
</префікс1:префікс2:елемент>
```

Оскільки вкладених просторів імен в XML не існує.

Ім'я разом з префіксом називається **розширеним** або **уточненим ім'ям (QName, Qualified Name)**. Частина імені, яка записана після двокрапки, називається **локальною частиною (local part)** імені.

Варто відмітити, що ідентифікатор простору імен повинен мати форму **URL** адреси різних Web-сайтів. Але це зовсім не означає, що ця адреса повинна існувати в мережі Internet. Програми, які використовують XML-документ, не звертаються по даній адресі. Просто ідентифікатор простору імен повинен бути дійсно унікальним і тому розробники технічної рекомендації по застосуванню просторів імен "Namespaces in XML" (http://www.w3c.org/TR/REC-xml-names) справедливо вирішили, що буде зручно використовувати для нього саме DNS-ім'я сайту, на якому розміщено визначення простору імен. Адже навряд чи комусь прийде в голову використовувати адресу чужого сервера в якості ідентифікатора свого простору імен. Ми ж дивимось на URI просто як на унікальний рядок символів, який ідентифікує простір імен. Зазвичай вказують URL фірми, яка створює дану реалізацію XML, або ж ім'я файлу з описом схеми XML.

Відносно двокрапки в імені елемента або атрибута, то по правилам SGML та XML, двокрапка може використовуватись в іменах і тому будь-яка програма, яка незнає просторів імен, розглядає дане уточнене ім'я як звичайне. Отже, в DTD-документі, якщо він існує, також не слід упускати префікси імен.

Наприклад:

```
<phone:phonebook xmlns:phone = "http://www.phone.com.ua/ns">
</phone:phonebook>
```

Атрибут **xmlns** може з'являтись в будь-якому елементі XML, а не лише в кореневому.

```
<phone:phonebook xmlns:phone = "http://www.phone.com.ua/ns">
    <info:person xmlns:info = "http://www.phone.com.ua/ns">
```

```
        ...
    </info:person>
</phone:phonebook>
```

В вищерозглянутому випадку, елементи phonebook та person мають різні префікси, але, не дивлячись на це, вони належать одному і тому ж розробнику або XML-схемі.

В одному елементі можна визначити кілька префіксів просторів імен. Як правило, такий елемент являється кореневим, що вказує на доступність і можливість використання цих префіксів в межах всього документа.

```
<phone:phonebook xmlns:phone="http://www.phone.com.ua/ns"
                 xmlns:vasja="http://www.VasjaPupkin.cool.ua">
</phone:phonebook>
```

Після того як префікс визначений, його можна використовувати для імен елементів та атрибутів, які ми хочемо віднести до простору імен http://www.phone.com.ua/ns. Наприклад:

```
<phone:city phone:type="смт">Zmerinka</phone:city>
```

Крім всього іншого, існує можливість визначати простір імен по замовчуванню (комбінація з іншими просторами також допускається). Такий простір імен має наступний синтаксис запису:

```
<phonebook xmlns="http://www.phone.com.ua/ns">
</phonebook>
```

Це надає можливість упускати префікс при описі елементів та атрибутів. В випадку появи в документі елемента без префікса, його ім'я належатиме простору імен по замовчуванню, якщо він визначений, інакше буде нулевим.

Але, слід відзначити, що атрибути ніколи не входять в простір імен по замовчуванню. Тобто, якщо атрибут вказаний без префікса, то це означає, що атрибут не відноситься ні до одного з визначених просторів імен і парсер не буде шукати такий атрибут в жодному з просторів.

Перепишемо XML документ, який ми написали на минулій парі, тобто записну книжку так, щоб вона використовувала простір імен.

notebook.xml

```
<?xml version="1.0" encoding="windows-1251"?>
<ntbk:notebook xmlns:ntbk="http://www.note.com/nt"
               xmlns:phone="http://www.phone.com.ua/ns" >
      <ntbk:person>
            <ntbk:name ntbk:first="Vasja" ntbk:surname="Pupkin"/>
            <ntbk:phone>22-22-22</ntbk:phone>
            <phone:city phone:type="смт">Zmerinka</phone:city>
      </ntbk:person>
      <ntbk:person>
            <ntbk:name ntbk:first="Dasha" ntbk:surname="Ivanova"/>
            <ntbk:phone>333-33-33</ntbk:phone>
            <phone:city>Kyev</phone:city>
      </ntbk:person>
</ntbk:notebook>
```

Підсумовуючи, варто сказати, що разом з специфікацією XML 1.1 W3C випустили специфікацію «Namespaces in XML 1.1». В основному її вихід пов'язаний з тим, що специфікації XML 1.0 та XML 1.1 відрізняються між собою. Нова версія специфікації притерпіла мінімальних змін. До цих змін варто віднести те, що в специфікації просторів імен для XML 1.0 можна було взагалі не оголошувати простори імен по замовчуванню, але не дозволялось не оголошувати окремий префікс. Цей факт приносив деяким розробникам незручності і тому в новій специфікації просторів імен в XML 1.1 для усунення цього недоліку передбачається рішення - префікс можна не оголошувати, асоціювавши його з пустим простором імен, наприклад: **xmlns:ntbk="".**

Але, не дивлячись на внесення таких змін в специфікацію для просторів імен, ці зміни не були внесені в інші специфікації XML, зокрема в специфікацію XML Schema. Наприклад, тип xml:string визначається на основі символів, допустимих в XML 1.0, але не в XML 1.1. Це означає, що насправді XML-схема не може бути використана для перевірки документів XML 1.1. Поки невідомо, як буде вирішуватись ця проблема, але консорціум знає про неї і займається її вивченням.

2. Поняття XML схем. Відмінності схем від DTD. Огляд існуючих XML схем

Разом з виходом першої версії XML для опису логічної структури XML-документа та правил його побудови, використовувались DTD-документи, які складались з набору формальних правил. Та цей набір був доволі примітивним та дуже скоро почав мало задовільняти розробників. Для прикладу, за допомогою DTD-документа Ви можете описати список наявних елементів та атрибутів, але водночас не можете накласти обмеження на те, скільки цих елементів може бути, типи елементів та атрибутів тощо.

В зв'язку з цим, з початку 1999 р. робоча група консорціуму W3C XML Schema Working Group розпочала роботу, а в травні 2001 р. запропонула новий засіб для опису структури документа XML - мову опису схем **XSD (XML Schema Definition Language)**. За допомогою цієї мови можна було записати схему XML (XML Schema), що описує всі конструкції, що використовуються в XML документі, включаючи детальний опис цих конструкцій: визначення використовуваних типів даних, кількості елементів і атрибутів, порядок їх слідування, вкладеність тощо. Додатковими **перевагами** XML схем над DTD документами являється те, що:

- XML схеми використовують синтаксис XML, завдяки чому не потрібно вивчати ще одну додаткову мову. Також для редагування можна використовувати будь-який XML-редактор, а для аналізу XML-парсер;
- XML схеми краще розширяємі, оскільки вони написані на XML. Завдяки цьому можна вбудовувати одні схеми в інші або ссилатись на інші схеми, створювати свої власні типи даних;
- XML схеми підтримують простори імен.

Мова XSD являється на сьогоднішній день самим найкращим рішенням щодо опису схем XML та стандартом електронної комерції, але вона не є єдиною. Наприклад, компанія Microsoft випустила свій власний стандарт **XDR (XML-Data Reduced)**. Згідно даного стандарту мова опису схем носить назву **XML-Data**, тобто XDR – це лише частина повної специфікації XML-Data. Варто відмітити, що компанія Microsoft випустила свій стандарт раніше стандарту W3C, але він не набув великого поширення і на сьогоднішній день стандарт XDR мало розповсюджений.

Крім XSD, до **найпоширеніших мов опису схем XML документів** на сьогоднішній день відносять:

- **Schematron** (http://www.ascc.net/xml/resource/schematron) – стандарт ISO, який був затверджений в 2006 році.
- **RELAX NG** (Regular Language Description for XML, New Generation, http://www.oasis-open.org/committees/relax-ng) – суміш мов Relax та TREX. Розробкою даного проекту керував Джеймс Кларк, а на сьогоднішній день її розвитком опікується технічний комітет групи OASIS (організація по просуванню стандартів для структурованої інформації).
- **Relax** (http://www.xml.gr.jp/relax).
- **TREX** (Tree Regular Expressions for XML, http://thaiopensource.cpm/trex).
- **DDML** (Document Definition Markup Language, http://purl.oclc.org/NET/ddml) – стандартизована в 1999 році і раніше відома як Xschema. Розроблена членами групи XML-dev, які працювали над забезпеченням функціональних можливостей DTD з використанням синтаксису XML. Після створення в консорціумі W3C XML Schema Working Group група XML-dev припинила своє функціонування, але стандарт DDML все ж був затверджений.

До **меньш розповсюджених**, разом з XDR належать такі мови як:

- **DCD** (Document Content Description) – стандартизована в 1998 році компаніями Microsoft та IBM. Вона об'єднує концепції XML-Data та синтаксис, відомий як RDF (Resource Description Framework – структура опису ресурсів);
- **SOX** (Schema for Object-Oriented XML) – стандартизована в 1998 році компанією VEO Systems, яку потім поглинула компанія CommerceOne.

Доречі доволі непоганий огляд мов опису схем представлений на сторінці http://www.oasis-open.org/cover/schemas.html. Ми з Вами розглянемо найбільш широко розповсюджену мову XSD. Адже, нам потрібно знати її плюси та мінуси.

3. Схеми XSD (стандарт W3C). Опис заголовка, елементів та атрибутів. Вбудовані типи даних XSD-схем

Отже, перейдемо безпосередньо до створення схем. Оскільки мова XSD являється найпоширенішою, то саме на ній ми і зробимо акцент та розберемо її синтаксичні конструкції.

Мова опису схем XSD створена як реалізація XML. Це означає, що XML схема записується у вигляді документа XML за виключенням того, що її елементи називаються **компонентами (components)**. Це зроблено для того, щоб відрізнити елементи схеми від елементів описуємого XML документа. Кореневий елемент схеми носить ім'я **schema**, а всі інші компоненти описують елементи XML документа та їх правила використання.

Варто відмітити, що імена елементів та атрибутів, які використовуються при описі XML схем, визначені в просторі імен з ідентифікатором http://www.w3c.org/2001/XMLSchema. Префікс імен, який відносять до даного простору імен, часто називають **xs** або **xsd**. Кожен аналізатор знає даний простір імен і розуміє імена з даного простору. Даний простір імен можна зробити також простором імен по замовчуванню, але тоді потрібно задати простір імен для визначаємих в схемі типів та елементів.

Отже, розпочнемо вивчення правил написання та побудови XSD схем. Враховуючи вищесказане, першим рядком нашої схеми буде вказання її кореневого елемента schema з визначенням простору імен:

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    ............
</xs:schema>
```

Як було вже також сказано, синтаксис XML схеми схожий з синтаксисом XML, тобто вона складається з компонентів (елементів) та атрибутів, які описують елементи та атрибути XML документа.

Для опису елемента, який буде входити до XML документа, в схемі використовується компонент **element**, який має наступний синтаксис застосування:

```
<xs:element name = "назва елемента"
            [type = "тип елемента"]
            [minOccurs = "найменша к-ть появ елемента в документі"]
            [maxOccurs = "найбільша к-ть появ елемента в документі"] />
```

Назва елемента повинна відповідати всім правилам іменування для XML-документів, які ми розглядали на першій парі, тому на цьому зупинятись не будемо.

Щодо **типу елемента**, то він дозволяє визначити характер даних, які можуть міститись в даному елементі: символьні дані, цілі або дробові числа, логічні значення тощо. Мова опису схем XSD містить багато вбудованих простих типів, які наведені в наступній таблиці.

Назва	Розмір	Характеристики
		Дійсні числа
decimal		Немає обмеження щодо розрядності (мантиси) числа, але вимагається, щоб можна було записати не менше 18 цифр
float		Дійсні значення з одинарною точністю
double		Дійсні значення з подвійною точністю
		Цілі числа
long	8	-9 223 372 036 854 775 808 .. +9 223 372 036 854 775 807
integer (int)	4	-2 147 483 648 .. +2 147 483 647
short	2	-32 758 .. + 32 767
byte	1	-128 .. +127
nonPositiveInteger		Підтип integer. Складається з недодатніх чисел з довільною кількістю цифр (0, -1,-2….)
negativeInteger		Підтип integer. Складається з від'ємних чисел з довільною кількістю цифр (-1,-2,-3…)
nonNegativeInteger		Підтип integer. Складається з невід'ємних чисел з довільною кількістю цифр (0,1,2…). Має наступні підтипи даних: • unsignedLong; • unsignedInt; • unsignedShort; • unsignedByte.
positiveInteger		Підтип integer. Складається з додатніх чисел з довільною кількістю цифр (1,2,3….)
		Рядкові типи
string		Довільний рядок символів
normalizedString		Підтип string, що не містить наступних символів: '\n', '\t' та '\r'. Фактично, перед обробкою схеми, ці символи перетворюються в пропуски.
token		Підтип normalizedString, в якому не містяться, крім всього іншого, початкові та кінцеві пробільні символи, а також розташовані підряд кілька пропусків. Має 3 підтипи даних: • language – запис назви мови згідно рекомендації RFC 1766 (ru, en, de, fr); • name - імена XML, що можуть містити літери, цифри та символи (:), (-), (.), (_). Ці імена повинні починатись з літери або знака (_); • NCName (Non-Colonized Name) – імена, які не містять двокрапку (:). Він має 3 підтипи: ID, ENTITY та IDREF.
		Дата та час
duration		Описує проміжок часу, наприклад, запис в форматі P1Y2M3DT10H30M45S означає один рік (1Y), два місяці (2M), три дні (3D), 10 годин (10H), 30 хвилин (30M) та 45 секунд (45S). Допускається скорочення запису, наприклад: P120M – 120 місяців, або T120M – 120 хвилин

dateTime		Містить дату в форматі **CCYY-MM-DDThh:mm:ss**, наприклад, 2006-07-12T10:30:02
time		Час в форматі **hh:mm:ss**
date		Дата в форматі **CCYY-MM-DD**
gYearMonth		Час та місяць в форматі **CCYY-MM**
gMonthDay		Місяць та день в форматі **–MM-DD**
gYear		Час в форматі **CCYY**
gMonth		Місяць в форматі **–MM-**
gDay		День в форматі **-DD**
Двійкові дані		
hexBinary		Шіснадцяткова система числення без додаткових символів, наприклад, 0B2F або 356C0A
base64Binary		Число в кодуванні Base64
Інші вбудовані прості типи		
anyURI		URI-адреси
QName		Розширене ім'я тега або атрибута (qualified name), тобто ім'я з префіксом, відділеним від імені двокрапкою
NOTATION		Використовується для запису математичних, хімічних та інших символів, нот, абетки Бройля та інших позначень

Необов'язкові атрибути **minOccurs** та **maxOccurs** визначають частоту появи даного елемента в XML документі. Значення по замовчуванню для них рівне 1. Це означає, що якщо ці атрибути відсутні, то елемент повинен з'явитись в документі XML рівно 1 раз. Значення атрибута minOccurs встановлене в 0 вказує на те, що елемент може бути взагалі відсутнім в документі, а maxOccurs з значенням **unbounded** визначає, що максимальна кількість появи елемента не обмежена.

Наприклад:

```
<xs:element name="rating" type="xs:positiveInteger" minOccurs="unbounded" />
```

Дане оголошення, дозволяє описати елемент, з назвою rating, який буде містити лише додатні числа, починаючи з 1. Даний елемент обов'язково повинен бути присутнім в документі мінімум 1 раз, а максимальна кількість появ не обмежена.

Зверніть також увагу на те, що при вказанні типу елемента обов'язково необхідно задавати простір імен:

```
xs:positiveInteger
```

Вказання типу елемента в атрибуті **type** зручне, якщо це вбудований простий тип або тип, описаний раніше. Тоді в атрибуті type можна описати лише ім.'я типу. Якщо ж тип елемента визначається на місці, то визначення типу елемента краще винести в вміст компонента element.

Наприклад:

```
<xs:element name="person">
  <!-- визначення типу елемента -->
</xs:element>
```

В вищеописаному прикладі, тип для елемента person не задається. Замість цього, в середині компонента element буде міститись опис всіх елементів та/або атрибутів, які будуть входити до його складу.

Оголошення атрибутів для XML документа схоже з оголошенням елемента та має наступний вигляд:

```
<xs:attribute name = "назва атрибута"
              [type = "тип атрибута"]
              [use = "обов'язковість атрибута"]
             [default = "значення по замовчуванню"]
              [fixed = "значення по замовчуванню"] />
```

Правила вказання **назви та типу атрибута** аналогічні правилам елемента, але існує ще кілька додаткових типів для атрибута, які включені для сумісності XML схем та XML 1.0 DTD. До цих типів відносять: IDREF, IDREFS, ENTITY, ENTITIES, NOTATION, NMTOKEN та NMTOKENS. Характеристики цих типів були розглянуті при вивченні DTD документів.

Необов'язковий атрибут **use** може приймати одне з наступних значень:

- **optional** – атрибут являється необов'язковим (значення по замовчуванню);
- **required** – атрибут являється обов'язковим;

- **prohibited** – атрибут заборонений. Дане значення корисно, якщо ви розробляєте, наприклад, ряд версій XSD схеми для опису XML документа. Припустимо, в ході розробки та випуску нової версії XML документа певні атрибути стали непотрібними. Якщо ці атрибути викинути з схеми, то при її підключенні користувачу буде згенерована помилка, яка вкаже на те, що цих атрибутів в XML документі не може бути. Виходом з цієї ситуації є встановлення таких атрибутів в значення prohibited, після чого для таких атрибутів буде просто викликатись попередження, наприклад, про те, що ці атрибути в новій версії не підтримуються і їх бажано не використовувати. Також значення prohibited корисно при визначенні підтипу, щоб відмінити деякі атрибути базового типу.

Якщо описуємий атрибут являється необов'язковим, то атрибутом **default** можна задати його значення по замовчуванню. Як видно з опису, значення по замовчуванню можна також задати атрибутом **fixed**, але його принцип дії дещо відрізняється від атрибута default. Значення, яке вказується в атрибуті fixed, буде єдиним значенням описуємого атрибута, тобто такий атрибут може приймати лише задане значення.

Наприклад:

```
<xs:attribute name="id"   type="positiveInteger" use="required"/>
<xs:attribute name="name" type="NCName" default="unknown"/>
```

Визначення типу атрибута, який повинен бути обов'язково простим типом, можна винести в вміст компонента attribute, подібно до розширення опису елемента. В такому випадку, опис атрибута набуде наступного вигляду:

```
<xs:attribute name="quantity">
  <!-- визначення типу атрибута -->
</xs:attribute>
```

4. Прості та комплексні типи даних

В XML схемах за допомогою вбудованих типів можна визначати свої власні нові типи елементів. Але, преш ніж перейти до їх створення, потрібно зрозуміти як мова схем XSD працює з елементами. Перш за все, вона поділяє всі елементи XML документа на **прості** і **складні**.

Простими (simple) елементами описуємого документа XML вважаються елементи, які не містять атрибутів та вкладених елементів. До простих типів даних слід віднести вбудовані типи (decimal, float, double, string, name тощо).

Складні (complex) елементи – це елементи, які можуть містити інші елементи і/або хоча б один атрибут.

Для кращого розуміння суті простих та складних типів, наведемо невеличкий приклад:

```
<name>Vasja</name>      <!-- простий тип -->

<person>    <!-- комплексний тип -->
      <name>Vasja</name>      <!-- простий тип -->
</person>

<person>    <!-- комплексний тип -->
      <name x="He-he">Vasja</name>  <!-- комплексний тип -->
</person>
```

Прості типи елементів описуються гкомпонентом **simpleType**, а складні типи – **complexType**.

Отже, щоб створити новий простий тип нам достатньо прописати наступне:

```
<xs:simpleType name="назва типу">
  <!-- визначення типу -->
</xs:simpleType>
```

В атрибуті name задається назва майбутнього простого типу, а в середині описується його визначення. Як правило, нові прості типи являються **звуженнями (restriction)** вже існуючих базових (вбудованих) або раніше оголошених простих типів, **списками (list)** або **об'єднаннями (union)**.

Щодо комплексного типу, то він визначається, як вже було вище сказано, компонентом complexType, який має наступний вигляд:

```
<xs:complexType name="назва типу">
  <!-- визначення типу -->
</xs:complexType>
```

Аналогічно оголошенню простого елемента, в атрибуті name задається назва майбутнього комплексного (складного) типу. В вмісті компонента complexType записуються елементи, які включаються в складний тип, або/і атрибути відкриваючого тега. Крім того, існує також можливість розширювати створений комплексний тип іншими атрибутами та елементами.

5. Звуження типів

Як ми вже раніше згадували, нові прості типи являються **звуженнями (restriction)** вже існуючих базових типів або раніше оголошених простих типів. Це зручно робити, якщо в XML документі існують певні елементи, значення яких необхідно обмежити. Так, маючи у використанні базовий тип xs:string, Ви можете обмежити його максимальною кількістю символів або ж задати маску (шаблон) на дані, що вводяться.

Звуження простого типу визначається компонентом **restriction (звуження)**. Базовий тип задається атрибутом **base**. Сам вміст даного компонента містить опис обмеження для вказаного базового типу.

Наприклад, виникла необхідність звузити базовий тип string, встановивши для нього наступні обмеження:

- максимальна кількість символів – 7;
- вводити можна лише символи або знак підкреслення.

```
<xs:simpleType name="codeType">
      <xs:restriction base="xs:string">
            <xs:length value="7" /> <!-- обмежуємо можливу довжину рядка: 7 символів -->
            <xs:pattern value="\w" /> <!-- шаблон на введене значення: лише символи -->
      </xs:restriction>
</xs:simpleType>
```

Елементи <length> та <pattern> власне і задають обмеження типу. Називаються вони **фасетками (facets)**. До фасеток відносять наступні елементи:

До них відносять наступні:

Назва фасетки	Опис
<maxExclusive>	Найбільше значення, яке більше не входить в визначаємий тип
<maxInclusive>	Найбільше значення визначаємого типу
<minExclusive>	Найменьше значення, яке більше не входить в визначаємий тип
<minInclusive>	Найменьше значення визначаємого типу
<totalDigits>	Загальна кількість цифр в визначаємому числовому типі; звуження типу decimal
<fractionDigits>	Кількість цифр в дробовій частині числа
<length>	Довжина значень визначаємого типу
<maxLength>	Найбільша довжина значень визначаємого типу
<minLength>	Найменьша довжина значень визначаємого типу
<enumeration>	Одне з перерахованих значень
<pattern>	Регулярний вираз
<whiteSpace>	Використовується при звуженні типу string та визначає спосіб приведення символів ’\n’, ‘\t’ та ‘\r’. Атрибут **value** даного елемента може приймати наступні значення: • preserve – не забирати escape-послідовності; • replace – замінити escape-послідовності пропусками; • collapse – після заміни escape-послідовностей пропусками забрати початкові та кінцеві пропуски, а замість кількох пропусків, які йдуть підряд залишити лише один.

В тегах-фасетках також можуть мати атрибути. Ці атрибути називають **базисними фасетками (fundamental facets)**. Серед них виділяють:

Базисна фасетка	Опис
ordered	Задає впорядкованість визначає мого типу та може приймати одне з наступних значень: - false – тип невпорядкований; - partial – тип частково впорядкований; - total – тип повністю впорядкований.
bounded	Задає обмеженість або необмеженість типу значенням true або false
cardinality	Задає скінченність або нескінченність типу значенням finite або countably infinite
numeric	Показує, чи значення являється числовим типом. Може приймати значення true або false

Підсумовуючи, можна сказати, що для звуження типу використовується компонент restriction, в середині якого встановлюються обмеження, що задаються фасетками. Фасеток може бути кілька, в залежності від необхідності. Крім того, фасетки можуть бути розширені базисними фасетками.

Після того, як Ви створили свій власний тип, його можна використати при описі елемента як і будь-який простий тип:

```
<xs:element name="code" type="codeType" />
```

Опис нового простого типу, можна також включити у вміст компонента element. В такому випадку, визначення елемента набуде вигляду:

```
<xs:element name="code">
  <xs:simpleType>
    <xs:restriction base="xsd:string">
      <xs:length value="7" />
      <xs:pattern value="\w" />
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Для закріплення матеріалу, розглянемо ще кілька прикладів створення власних простих типів, шляхом звуження базових:

```
<xs:simpleType name="Year">
  <xs:restriction base="xs:gYear">
    <xs:minInclusive value="1700"/> <!-- мінімальне введенне значення року - 1700 -->
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="DateType">
  <xs:restriction base="xs:date">
    <xs:minInclusive value="2003-01-01"/> <!-- мінімальна дата -->
    <xs:maxInclusive value="2003-12-31"/> <!-- максимальна дата -->
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="TypeId">
  <xs:restriction base="xs:integer">
    <xs:minInclusive value="10000" />      <!-- мінімальне значення >= 10 000 -->
    <xs:pattern value="[1-7][0-7]{4}" />   <!-- шаблон на введення -->
  </xs:restriction>
</xs:simpleType>

<xs:simpleType name="genderType">
  <xs:restriction base="xs:string">
    <xs:pattern value="male|female"/>   <!--шаблон на введення одного з двох значень-->
  </xs:restriction>
</xs:simpleType>
```

Якщо необхідно створити елемент, який буде містити наперед визначені значення, тоді також необхідно скористатись звуженням певного типу. Для встановлення переліку можливих значень використовується фасетка <enumeration>.

```
<xs:attribute name="size">
  <xs:simpleType>
    <xs:restriction base="xs:token">
      <xs:enumeration value="small" />
      <xs:enumeration value="medium" />
      <xs:enumeration value="large" />
    </xs:restriction>
  </xs:simpleType>
</xs:attribute>
```

Після цього описаний атрибут в XML-документі можна використовувати наступним чином:

```
<header size="small">Розділ 1</header>
```

6. Списки та об’єднання

Список – це простий тип елементів, в тілі якого записуються через пропуск кілька значень одного і того ж простого типу. Наприклад, в XML документі може зустрітись елемент, який містить список символьних значень, що вказує характеристики шрифта:

```
<font-effects>Italic Underline</font-effects>
```

Для опису такого XML елемента в XSD схемі використовується компонент **list**. Тип елементів списка можна вказати або в атрибуті **itemType**, або визначити в вмісті самого елемента list. Наприклад, тип списка для елемента XML документа, приклад якого ми навели, буде описаний наступним чином:

```
<xs:simpleType name="listFontEffects">
  <xs:restriction>

    <xs:simpleType>
      <xs:list itemType="xs:string" />
    </xs:simpleType>

    <xs:maxLength value="12" />
  </xs:restriction>
</xs:simpleType>
```

В нашому прикладі, ми створили тип з назвою listFontEffects, який вказує на те, що це буде список рядкових значень, максимальною довжиною 12 символів.

У випадку, якщо елемент буде складатись з різнотипних значень, компонент list нам не підійде. Для такого опису використовується простий тип **об'єднання**, який визначається компонентом **union**. В його атрибуті **memberTypes** можна вказувати імена типів, які планується об'єднати.

Наприклад:

```
<xs:union memberTypes="xs:string xs:integer listFontEffects" />
```

Другий спосіб оголошення об'єднання – це записати опис простих типів в вмісті компонента union. Наприклад:

```
<xs:simpleType name="FontEffectType">
  <xs:union>

    <xs:simpleType>
      <xs:restriction base="xs:integer">
        <xsd:minInclusive value="1"/>
        <xsd:maxInclusive value="4"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType>
      <xs:restriction base="xs:token">
        <xs:enumeration value="Bold" />
        <xs:enumeration value="Italic" />
        <xs:enumeration value="Underline" />
        <xs:enumeration value="Strikeout" />
      </xs:restriction>
    </xs:simpleType>

  </xs:union>
</xs:simpleType>
```

Після цього елемент XML-документа font-effect, який має описаний тип, може описуватись одним з вказаних нижче способів:

```
<font-effect>Bold</font-effect>
<font-effect>3</font-effect>
```

7. Визначення складних типів. Моделі групи. Розширення та звуження складних типів

Нагадаємо, що комплексним або складним типом, являється елемент, який містить дочірні елемента або/і хоча б один атрибут. Для оголошення комплексного типу використовується конструкція **complexType**, у вмісті якої задається визначення самого типу.

Визначення комплексного типу можна поділити на **4 групи**:

1. визначення типу пустого елемента, тобто елемента, який має лише атрибути;
2. визначення типу елемента, який містить вкладені елементи;
3. визначення типу елемента, який містить лише текст.

Причому, кожна з трьох останніх груп може містити атрибути.

Розглянемо всі ці групи по-порядку. Почнемо з першої, тобто якщо необхідно **описати пустий елемент**, який містить лише атрибути. Прикладом такого елемента може бути елемент **fullname**, який описує повне ім'я особи. Припустимо, що ім'я та прізвище задаються за допомогою атрибутів firstname і lastname.

```
<fullname firstname="Olga" lastname="Ivanova" />
```

В такому разі в XSD схемі опис такого елемента буде наступним:

```
<xs:element name="fullname">

  <xs:complexType>
    <xs:attribute name="firstname" type="xs:string" />
    <xs:attribute name="lastname" type="xs:string" />
  </xs:complexType>

</xs:element>
```

або

```
<xs:element name="fullname" type="personInfo" />

<xs:complexType name="personInfo">
  <xs:attribute name="firstname" type="xs:string" />
  <xs:attribute name="lastname" type="xs:string" />
</xs:complexType>
```

До другої групи відносять елементи, які можуть містити вкладені елементи. Наприклад, елемент person може містити дочірні елементи address, phone тощо.

```
<person>
  <address>Вінницька обл., м.Жмеринка, вул.Кривоноса, 45/256</address>
  <phone>22-22-22</phone>
</person>
```

Якщо **складний тип містить дочірні елементи**, тоді їх необхідно перелічити, вказавши при цьому їх назви та послідовність. Але перед тим, як перераховувати їх описи, потрібно обрати **модель групи (model group)** вкладених елементів, оскільки вони можуть з'являтись в довільному або певному визначеном порядку. Саме модель групи і дозволяє це вказати.

Модель групи визначається одним з **трьох компонентів**:

1. **sequence** – перераховані елементи повинні записуватись в документі в визначеному порядку;
2. **all** - перераховані елементи можуть записуватись в документі в довільному порядку;
3. **choice** – можна встановлювати лише один елемент з перерахованих.

Наприклад:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence> <!-- <xs:all> | <xs:choice> -->
      <xs:element name="address" type="xs:string" />
      <xs:element name="phone" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

При використанні першої моделі групи (**sequence**) ми вказуємо обов'язковість та чіткий порядок появи вкладених елементів в елементі person: спочатку елемент address, а потім phone.

Маючи справу з моделлю групи **all**, Ви можете спочатку вказати телефон особи, а потім її адресу і навпаки.

Якщо ж обрати модель вибору **choice**, то при описі елемента person в XML документі, ви можете вказати або її адресу, або телефон. Для нашого прикладу така модель групи не підійде, але така модель групи себе може виправдати в іншому випадку. Наприклад, при описі особи замість одного телефону (phone), необхідно вказати список телефонів, які задаються

елементом phone-list, або ж при вказанні імені надати вибір на вказання повного імені (fullname) чи лише прізвища (surname). Для такого випадку опис елемента person дещо зміниться:

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>

      <xs:element name="address" type="xs:string" />
      <xs:choice>
        <xs:element name="phone" type="xs:string" />
        <xs:element name="phone-list">
          <xs:simpleType>
            <xs:list itemType="xs:string" />
          </xs:simpleType>
        </xs:element>
      </xs:choice>

    </xs:sequence>
  </xs:complexType>
</xs:element>
```

Як видно з прикладу, компонент **choice** можна вкладати в компонент **sequence** або навпаки. Причому, ми можемо вказати для будь-якого компонента моделі групи атрибут **maxOccurs = "unbounded"**, що дозволить повторювати групу довільну кількість разів.

Але модель групи **all** відрізняється в даному плані від моделей **sequence** та **choice**. В ній не допускається використання компонентів **sequence** та **choice**. Аналогічно, в компонентах **sequence** і **choice** не можна використовувати компонент **all**.

Якщо в елементі person будуть міститись атрибути, то їх опис слід винести за межі опису моделі групи.

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="address" type="xs:string" />

      <xs:choice>
        <xs:element name="phone" type="xs:string" />
        <xs:element name="phone-list">
          <xs:simpleType>
            <xs:list itemType="xs:string" />
          </xs:simpleType>
        </xs:element>
      </xs:choice>

    </xs:sequence>

    <xs:attribute name="firstname" type="xs:string" />
    <xs:attribute name="lastname" type="xs:string" />
  </xs:complexType>
</xs:element>
```

Такий елемент в XML документі може описуватись наступним чином:

```
<person firstname="Вася" lastname="Пупкін">
  <address>Вінницька обл., м.Жмеринка, вул.Кривоноса, 45/256</address>
  <phone>22-22-22</phone>
</person>
```

Наведемо ще кілька прикладів опису комплексних типів з вмістом:

```
<xs:complexType name="AddressType">
  <xs:all>
    <xs:element name="street" type="xs:string"/>
    <xs:element name="city" type="xs:string" minOccurs="0"/>
    <xs:element name="state" type="State" minOccurs="0"/>
  </xs:all>
</xs:complexType>
```

```
<xs:complexType name="PublicationsListType">
  <xs:choice maxOccurs="unbounded">
    <xs:element name="book" type="xs:string"/>
    <xs:element name="article" type="xs:string"/>
    <xs:element name="whitepaper" type="xs:string"/>
  </xs:choice>
</xs:complexType>
```

Опис елемента, який містить текст дещо відрізняється від попередніх і дещо складніший. Тому, перед тим, як до нього перейти, варто розглянути поняття розширення типів.

В попередніх розділах було розглянутий механізм звуження базових типів, який здійснювався за допомогою конструкції restriction. Крім звуження типів, дозволяється їх розширювати, але це дозволено лише для комплексних типів. Розширення типу використовується тоді, коли нам необхідно додати до вже описаного типу додаткові атрибути чи елементи. Наприклад, існує новий комплексний тип, що описує адресу особи: місто та телефон. В ході роботи знадобився ще один тип, який описував би повну адресу, яка б включала б і назву вулиці. В такому випадку можна написати новий тип для елемента fulladdress, а можна взяти раніше описаний тип address і розширити його вулицею. Після цього на базі розширення можна створити новий комплексний тип.

Для розширення комплексного типу використовується компонент **complexContent**, який містить компонент **extension (розширення)**. В атрибуті **base** останнього вказується базовий тип, тобто тип, який потрібно розширити. Але тепер це обов'язково повинен бути складний, а не простий тип.

Для ілюстрації використання розширення, скористаємось описаним вище прикладом з адресами:

```
<xs:complexType name="AddressType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="city" type="xs:string" />
    <xs:element name="phone" type="xs:string" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="FullAddressType">
  <xs:complexContent>
    <xs:extension base="AddressType">
      <xs:sequence>
        <xs:element name="street" type="xs:string" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Крім розширення комплексних типів, їх також можна звужувати. Для цього в компоненті complexContent вказується компонент **restriction**. В його вмісті вказується перелік тих елементів, які залишаться після звуження. Наприклад, існує тип bookType, який описує характеристики книги і необхідно створити новий складний тип, який буде містити аналогічну інформацію, але без вказання інформації про рік видання:

```
<xs:complexType name="bookType">
  <xs:sequence maxOccurs="unbounded">
    <xs:element name="author" type="xs:normalizedString" minOccurs="0"
                                                         maxOccurs="unbounded" />
    <xs:element name="title" type="xs:string" />
    <xs:element name="year" type="xs:date" minOccurs="0" maxOccurs="1" />
  </xs:sequence>

  <xs:attribute name="id" type="xs:integer" use="optional" />
</xs:complexType>

<!-- звужений тип -->
<xs:complexType name="bookExtensionType">
  <xs:complexContent>
    <xs:restriction base="bookType">
      <xs:sequence>
        <xs:element name="author" type="xs:normalizedString" minOccurs="0"
                                                             maxOccurs="unbounded" />
        <xs:element name="title" type="xs:string" />
```

```
      </xs:sequence>
      <xs:attribute name="id" type="xs:integer" use="optional" />
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
```

Підсумовуючи, слід відзначити, що в мову схем XSD внесено ряд елементів ООП (зачіпати ми їх не будемо), в зв'язку з чим розширений або звужений тип пов'язані із своїм базовим типом відношеннями успадкування, і до них можна застосувати операцію підстановки. У всіх типів мови XSD є спільний батько – базовий тип **anyType**. **Він являється батьківським для всіх складних типів**. Від типу anyType походить тип **anySimpleType - спільний предок для всіх простих типів**. Це дещо нагадує мови програмування, наприклад, в C# і Java базовим класом для всіх являється клас Object, а в MFC – CObject. Таким чином, всі складні типи визначаються як звуження типу anyType, а прості типу anySimpleType.

Розглянувши розширення типу, познайомимось з тим як описувати елемент, який містить простий текст. Для цього в тілі компонента complexType вводиться компонент **simpleContent**, в якому вказується або компонент **restriction**, або **extension**. В атрибуті **base** останніх задається простий тип, який описує тип тіла елемента. Виглядає це наступним чином:

```
<xs:complexType name="phoneType">
  <xs:simpleContent>
    <xs:restriction base="xs:string">
      <xs:pattern value="[1-9][0-9]-[0-9]{2}-[0-9]{2}" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

Таким чином ми описали тип, тіло якого містить значення вбудованого простого типу xs:string, який звужений маскою на введене значення.

Якщо необхідно додати атрибути до типу, то опис зміниться на наступний:

```
<xs:complexType name="phoneType">
  <xs:simpleContent>
    <xs:restriction base="xs:string">
      <xs:pattern value="[1-9][0-9]-[0-9]{2}-[0-9]{2}" />

      <xs:attribute name="code" type="xs:nonNegativeInteger" />
    </xs:restriction>
  </xs:simpleContent>
</xs:complexType>
```

Ми використали компонент restriction, оскільки використовували фасетки для звуження типу, якщо звуження типу не робити, то можна говорити про розширення типу атрибутом.

Наприклад:

```
<xs:complexType name="phoneType">
  <xs:simpleContent>
    <xs:extension base="xs:string">
      <xs:attribute name="code" type="xs:nonNegativeInteger" />
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
```

8. Посилання на елементи

Для спрощення використання елементів, в XSD схемах використовують **посилання**. Це дозволяє описати елементи в довільному місці схеми один раз і ссилатись на них в випадку необхідності. Це особливо корисно, коли комплексний елемент, опис якого доволі громіздкий, зустрічається при описі схеми багато разів. Наприклад, елемент name або address тощо. Для ссилки на раніше описаний елемент використовується конструкція **ref**, в якості значення якої вказується назва елемента, на який Ви посилаєтесь.

Наприклад:

```
<xs:element name="name" type="xs:string" />

<xs:element name="person">
<xs:complexType>
```

```
      <xs:sequence>
            <xs:element ref="name"/>
            <xs:element name="address" type="xs:string" minOccurs="0" />
      </xs:sequence>
</xs:complexType>
</xs:element>
```

Крім того, ми маємо можливість ссилатись на свій власний елемент та робити його вкладеним, але тоді є обов'язковим задання параметру **minOccurs** рівним 0, що вказує на те, що даний елемент може бути відсутнім.

```
<xs:element name="person">
<xs:complexType>
      <xs:sequence>
            <xs:element ref="name"/>
            <xs:element name="address" type="xs:string" minOccurs="0" />

            <xs:element ref="person" minOccurs="0" />
      </xs:sequence>
</xs:complexType>
</xs:element>
```

9. Анотації

Мова XML схем XSD підтримує анотації, які дозволяють задати короткий опис самої схеми, а також окремих елементів її вмісту для зручності її розуміння в подальшому. Говорячи іншими словами, ми можемо при описі XML схеми писати коментарі, які призначені для читання як людиною, так і прикладним додатком, який буде обробляти Вашу схему.

Мова XSD підтримує **три компоненти**, які призначені для аннотації схеми:

1. **annotation** – основний компонент, в якому описується анотація до схеми. Він включає в себе два нижчеописаних компоненти.
2. **documentation** – компонент, який власне і містить опис схеми, інформацію про авторські права тощо. Даний компонент може мати атрибут **xml:lang**, в якому вказується мова написання самої анотації. Доречі, атрибут xml:lang можна також задавати в елементі schema, для того, щоб вказати мову всієї схеми.
3. **appInfo** – компонент, який може використовуватись для представлення інформації про інструментальні засоби розробки, таблиці стилів та інших додатків. Наприклад, в середині компонента appInfo можна представити інформацію про те, які фасетки можуть бути застосовані до кожного простого типу. Потім, цю інформацію можна використати спеціальним додатком для автоматичної генерації тексту XML документа.

Компонент annotation зазвичай розміщують на початку схеми, але його також можна використовувати і перед описом нових типів, елементів чи атрибутів. Наприклад:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
           attributeFormDefault="unqualified">
  <xs:element name="root">
    <xs:annotation>
      <xs:documentation>Дана схема призначена для опису складних типів
      </xs:documentation>
    </xs:annotation>

    <xs:complexType>
      <xs:sequence>
        <xs:annotation>
          <xs:documentation>Елемент, який зберігає інформацію про адресу
          </xs:documentation>
        </xs:annotation>
        <xs:element name="address" type="AddressType" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:annotation>
    <xs:documentation>Складний тип для опису інформації про адресу</xs:documentation>
  </xs:annotation>
  <xs:complexType name="AddressType">
```

```
    <xs:sequence maxOccurs="unbounded">
      <xs:element name="city" type="xs:string" />
      <xs:element name="phone" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

10. Підключення XML схеми до документа

А тепер розглянемо питання про те, як підключити створену схему до нашого документа. Адже парсеру (синтаксичному аналізатору), який перевіряє відповідність XML документа схемі, потрібно якось вказати файли, які містять схему документа.

Для цього в документі XML, в кореневому елементі потрібно вказати шлях до схеми. Це можна зробити одним з **двох способів**:

1. В випадку, якщо елементи XML документа не належать жодному простору імен і записані без префікса, то в корневому елементі документа записується атрибут **noNamespaceSchemaLocation**, який вказує місце розташування файла з схемою в формі URL:

```
<notebook xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance"
          xsi:noNamespaceSchemaLocation="notebook.xsd">
```

2. В випадку, якщо елементи XML документа належать певному простору імен, то використовується атрибут **schemaLocation**, в якому через пропуск перераховуються простори імен та розміщення файла з схемою, яка описує даний простір імен. Наприклад:

```
<notebook xmlns ="http://www.VasjaPupkin.org/ns"
          xmlns:xsi="http://www.w3c.org/2001/XMLSchema-instance"
          xsi:schemaLocation ="http://www.VasjaPupkin.org/anyNS a.xsd
                               http://www.Olga.org/otherNS b.xsd"
          xmlns:vasja = "http://www.VasjaPupkin.org/anyNS"
          xmlns:olga = "http://www.Olga.org/otherNS" >
```

11. Приклади створення XSD схем

Наведемо кілька прикладів створення XSD схем, разом з XML документами, які використовують описані в них правила побудови.

1. Приклад записної книги, описаної на попередніх парах: sources\notebook
2. Приклад оформлення замовлення на покупку: sources\purchaseOrder
3. Приклад оформлення звіту про виконання досліду: sources\task

12. Домашнє завдання

1. До створеного на попередніх парах XML документа (книга рецептів) напишіть XSD-схему. Додати також анотацію, яка описує призначення схеми та інформацію про авторські права, а також мінімум дві анотації до елементів, атрибутів або нових типів даних. При описі схеми обов'язково скористатись посиланнями.

2. Створити нові типи, які дозволять описати наступні прості елементи XML-документа:
   - e-mail або URL адресу;
   - дійсне число в мові програмування C/C++. Наприклад: +0.5, .6, 89. , -2.e+5, +2.E5, -2.f тощо.

3. Написати XML документ «Академія», що має наступну структуру:
   1) Групи, що мають наступні властивості (атрибути):
      - назва групи, яка формується по наступному принципу:
        - перша цифра вказує на номер потоку;
        - далі йде абревіатура форми навчання (С або НС);
        - наступні цифри вказують на початок годин гавчання;
        - дефіс;
        - номер групи;
        - скорочена назва спеціальності.

        Наприклад, 7С8-1пр або 7НС8-1гр.

      - спеціалізація (може приймати одне з трьох значень: адміністрування, програмування, дизайн).

До складу групи входять студенти, кожен з яких має:
- повне ім'я (атрибути: імя, прізвище та по-батькові);
- повна адреса (дочірні елементи: країна, місто, вулиця, дім, телефон (регулярний вираз), email (регулярний вираз));
- розмір стипендії (максимум 500 грн.).

Передбачити, щоб ПІБ не містило недопустимих символів (цифр, *, _, -, тощо).

2) Предмети, кожен з яких містить дані про:
   - назву предмету;
   - план курса (назва прикріпленого документа з описом плану; можна з шляхом).

3) Викладачі, про кожен з яких зберігається наступна інформація:
   - повне ім'я (атрибути: імя, прізвище та по-батькові);
   - список предметів, які він читає;
   - кількість груп, в яких він читає (не може бути більше 10 або від'ємним).

Написати також XSD схему до вищеописаного XML документа. До схеми включити коротку анотацію про її роботу.