Web-технології та web-дизайн

2024/2025

Лекція №13
XML (частина 3)
XML Schema. XSLT.

Лекції: ст. викладач каф. Штучного інтелекту
Гриньова Олена Євгенівна
olena.hrynova@nure.ua

XML Schema - альтернатива використанню DTD

Мова опису схем

DTD (Document Type Definition) - мова визначення типу документів.

XDR (XML Data Reduced) – діалект XML, розроблений Microsoft.

XSD (**X**ML **S**chema **D**efinition Language) – консорціум W3C.

XSD – альтернатива DTD

Синтаксис DTD недостатньо гнучкий, через це для деяких задач DTD недостатньо виразний.
У xml-документах застосовується один синтаксис, а у DTD – інший.
Моделі вмісту та оголошення списків атрибутів важко читати та розуміти, розчаровує неможливість задання шаблонів даних в елементах та атрибутах.

З цих причин було запропоновано низку альтернатив для DTD. До них належить XML Schema (іноді називають XSchema).

Консорціум W3C виробив рекомендацію мови визначення схем XML (XSD), поєднавши найпопулярніші мови опису схем в один стандарт. Основна мета, яка при цьому ставилася, - отримання стандарту, який можна широко реалізувати, і при цьому він є **платформо-незалежним**.

XSD

За допомогою **XML-схем** можна не тільки визначити синтаксис документа (як у разі застосування визначень DTD), але й **задати** фактичні **типи даних вмісту** кожного елемента, **успадковувати** синтаксис інших схем, **анотувати** схеми, використовувати їх з численними просторами імен, створювати прості і складові типи даних.

Також можна визначити мінімальну та максимальну **кількість появи елемента** в документі, створювати **типи списків** та **групи атрибутів**, обмежувати **діапазони значень**, які можуть містити елементи, обмежувати спадкування властивостей схемами, об'єднувати фрагменти різних схем, вимагати, щоб атрибут або значення елемента були унікальними, а також багато іншого.

Документ XML вважається валідним, якщо він не тільки правильно сформований, але й відповідає XSD, яка вказує, які теги використовуються, які атрибути можуть містити ці теги, та які теги можуть зустрічатись, в тому числі, всередині інших тегів.

Схеми XSD, переваги

- XSD можна розширювати для майбутніх доповнень.
- XSD є багатшим та потужнішим, ніж DTD.
- XSD написаний на **XML**.
- XSD підтримує **типи** даних.
- XSD підтримує **простори імен**.
- XSD - це **рекомендація W3C**.

Список деяких функцій XSD

Схеми XSD здатні вирішувати такі завдання:

- Перерахування елементів в документі XML і перевірка наявності в документі тільки оголошених елементів.
- Оголошення і визначення атрибутів, які модифікують елементи документа.
- Визначення батьківсько-дочірніх відносин між елементами.
- Визначення стану і моделей вмісту для елементів та атрибутів.
- Задавання простих и складних типів даних.

XSD, приклад

У файлі **.xsd** міститься XML-схема. Зверніть увагу на використаний елементами схеми простір імен **xsd**, який відповідає "http://www.w3.org/2001/XMLSchema“. Тут оголошується елемент **document**.

Лістинг **xsd**

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
targetNamespace = "http://starpowder"
xmlns:ch05 = "http://starpowder"
elementFormDefault = "qualified">
<xsd:element name="document"></xsd:element>
</xsd:schema>
```

Лістинг test.**xml**

```
<?xml version="1.0"?>
<ch05:documents xmlns:ch05 = 'http://starpowder'>
  <ch05:data/>
</ch05:documents>
```

XSD, приклад

Елемент **documents** використовується в XML-документі разом з відповідним простором імен <ch05:documents>.

Зверніть увагу на те, що XML-документ також містить елемент **<ch05:data/>,** якого **немає** в **Схемі**. Це неправильно, тому буде повідомлення про помилку.

Написання XML-схеми

XSD зберігається в окремому документі файл.xsd, а потім схему можна зв'язати з XML-документом, щоб використовувати його.

schema.xsd

```
<?xml version = "1.0"?>
<xsd:schema xmlns:xsd = "http://www.w3.org/2001/XMLSchema"
  targetNamespace = "http://www.tutorialspoint.com"
  xmlns = "http://www.tutorialspoint.com" elementFormDefault = "qualified">
    <xsd:element name = 'class'>
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name = 'student' type = 'StudentType' minOccurs = '0'
          maxOccurs = 'unbounded' />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name = "StudentType">
    <xsd:sequence>
      <xsd:element name = "firstname" type = "xsd:string"/>
      <xsd:element name = "lastname" type = "xsd:string"/>
      <xsd:element name = "nickname" type = "xsd:string"/>
      <xsd:element name = "marks" type = "xsd:positiveInteger"/>
    </xsd:sequence>
    <xsd:attribute name = 'rollno' type = 'xsd:positiveInteger'/>
  </xsd:complexType>

</xsd:schema>
```

Елемент <Schema>

schema- це кореневий елемент **XSD**, та він **обов'язковий!**

<**xsd**:**schema** xmlns:**xsd** = "http://www.w3.org/2001/XMLSchema">

У рамках кореневого елемента схеми XSD, атрибутом **xmlns** визначається простір імен XMLSchema, який містить елементи й атрибути XSD схеми. Наведений вище фрагмент вказує, що елементи та типи даних, які використовуються в схемі, визначені в просторі імен http://www.w3.org/2001/XMLSchema, і ці елементи/типи даних **повинні мати** префікс **xsd:** . Це **обов'язково!**

targetNamespace = "http://www.tutorialspoint.com"
Наведений вище фрагмент вказує, що елементи, які використовуються в цій схемі, визначені в просторі імен http://www.tutorialspoint.com.
Це необов'язково.

xmlns = "http://www.tutorialspoint.com"
Наведений вище фрагмент вказує, що простір імен **за замовчуванням** - http://www.tutorialspoint.com.

Елемент <Schema>

elementFormDefault = "qualified"

Наведений вище фрагмент вказує, що будь-які елементи, які оголошені в цій схемі, мають бути уточнені простором імен, перед використанням їх в будь-якому XML-документі. Це **необов'язково**.

Referencing Schema (підключення schema.xsd в xml-файлі)

```
<?xml version = "1.0"?>
<class xmlns = "http://www.tutorialspoint.com"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://www.tutorialspoint.com/schema.xsd">

  <student rollno = "393">
    <firstname>Dinkar</firstname>
    <lastname>Kad</lastname>
    <nickname>Dinkar</nickname>
    <marks>85</marks>
  </student>
    …
  <student rollno = "593">
    <firstname>Jasvir</firstname>
    <lastname>Singh</lastname>
    <nickname>Jazz</nickname>
    <marks>90</marks>
  </student>
</class>
```

файл.xml

Referencing Schema

XML-документ, який перевіряється за допомогою схеми, також повинен мати **оголошення простору імен**. Простір імен завжди вказують в кореневому елементі екземпляру документа (**class**) за допомогою атрибута **xmlns.**
Це **обов'язково.**

xmlns = "http://www.tutorialspoint.com"
Наведений вище фрагмент визначає оголошення простору імен **за замовчуванням**. Цей простір імен використовуються валідатором Схеми для перевірки того, що всі елементи є частинами цього простору імен.

xmlns:**xsi** = "http://www.w3.org/2001/XMLSchema-instance"
Цей простір імен містить **елементи** й **атрибути** XMLSchema, які можна включати в документ XML. Загалом прийнято, що префікс **xsi** (XMLSchema-instance) використовуються для цього простору імен та додається на початку імен усіх елементів та атрибутів, які належать простору імен, відділяючись від них двокрапкою.

Referencing Schema

Після визначення **xsi** використовуйте атрибут **schemaLocation** (посилання на конкретну схему). Цей атрибут має два значення, простір імен та розташування XML-схемы (поділяються пробілом).

```
xsi:schemaLocation = "http://www.tutorialspoint.com/schema.xsd"
```

Абстрактна модель даних XSD

Схема XSD - це набір **компонентів** схеми.
Компонент схеми (Schema component) - це загальний термін для будівельних блоків, які складають з себе абстрактну модель даних схеми. Існує декілька видів компонентів схеми, які можна розділити на три групи.

Основні компоненти схеми, які **можуть** (визначення типів) або **повинні мати** (визначення елементів і атрибутів):

- Simple type definitions
- Complex type definitions
- Attribute declarations
- Element declarations

Абстрактна модель даних XSD

Secondary schema components наступні:

- Attribute group definitions
- Identity-constraint definitions
- Type alternatives
- Assertions
- Model group definitions
- Notation declarations

The "helper" schema components надають невеликі частини інших компонентів схеми, вони залежать від свого контексту:

- Annotations
- Model groups
- Particles
- Wildcards
- Attribute Uses

Компоненти схеми

Процес створення схеми включає в себе два етапи:
1) декларування елементів або атрибутів;
2) оголошення типів елементів або типів атрибутів.

Елементи й атрибути XML-документа оголошуються **компонентами** схеми XSD - **xsd:element** та **xsd:attribute**.

Абстрактна модель надає два види типів компонентів (**type definition**): **прості** (**simple type**) та **складні (complex types**).

Структура **XML-документа**, визначається компонентами схеми **xsd:simpleType** та **xsd:complexType**.

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

Компоненти, що застосовуються в схемах

Тип	Опис
All	Дозволяє відображати елементи групи в довільному порядку всередині елемента, який їх вміщає в собі
Annotation	Створює анотацію
Any	Дозволяє будь-якому елементу з даного простору (просторів) імен бути присутнім всередині вміщувального елемента послідовності або альтернативи
AnyAttribute	Дозволяє будь-якому атрибуту з даного простору (просторів) імен бути присутнім всередині вміщувального елемента compiexType або attributeGroup
Appinfo	Визначає інформацію, що використовується додатками в елементі
annotationAttribute	Створює атрибут
AttributeGroup	Групує оголошення атрибутів таким чином, що вони можуть використовуватись як група для визначенні складених типів
Choice	Дозволяє одному и тільки одному елементу, що міститься в групі, бути присутнім у вміщувальному елементі
ComplexContent	Містить розширення або обмеження на складений тип, який має перемішений вміст або лише елементи
ComplexType	Визначає складений тип, що підтримує атрибути та вміст елемента
Documentation	Містить текст, що передається користувачеві в елементі анотації
Element ….	Створює елемент …….

XSD - Simple Types

Модель контенту простого типу, яка визначає XSD - це **набір обмежень** для рядків та інформація про кодовані ними значення, що застосовують до нормалізованого значення інформаційного компонента **attribute** або інформаційного компонента **element** без дочірніх елементів.

XSD - Simple Types

Неформально це застосовується до **значень атрибутів** та **текстовому вмісту елементів**:

- **Простий елемент (без дочірніх елементів)**

Простий елемент може містити **тільки текст**. Він не може містити ніякі інші елементи.

<xsd:**element** name = "element-name" type = "xsd:element-type"/>

- **Атрибут**

Атрибут сам по собі є типом і використовується в складному елементі.

<xsd:**attribute** name = "attribute-name" type = "xsd:attribute-type"/>

- **Обмеження**

Обмеження визначається допустимими значеннями компонента XML.

<xsd:**restriction** base = "xsd:element-type"> *restrictions* </xsd:restriction>

XSD - Simple Types

Існують дві головні категорії моделі контенту компоненти простого типу :

- **вбудовані** типи;
- **визначені користувачем прості** типи.

Мова XSD має велику кількість **вбудованих простих типів** даних. Вбудовані типи включають в себе **примітивні** й **похідні типи**.

Примітивні типи даних не створені з інших типів даних. Наприклад, числа с плаваючою комою - математичне поняття, яке не створено з інших типів даних.

Похідні типи даних визначені в термінах існуючих типів даних. Наприклад, ціле число - окремий випадок, створений з десяткового типу даних.

Іменовані обмеження називаються **фасетами** (Facets). Facets обмежують допустимі значення простих типів.

XSD - Simple Types

Значення за замовчуванням та фіксовані значення для **простих елементів**

Прості елементи можуть мати **значення за замовчуванням** або задане **фіксоване** значення.

Якщо не вказане інше значення, елементу автоматично присвоюється значення за замовчуванням .

<**xsd:element** name="color" type="xsd:string" **default**="red"/>
<**xsd:element** name="form" type="xsd:string" **fixed**="circle"/>

XSD - Simple Types

Обов’язкові й необов’язкові атрибути

За замовчуванням атрибути не є обов’язковими. Щоб вказати, що атрибут є **обов’язковим**, використовуйте параметр **use="required"**:

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

use - необов’язковий параметр, він визначає те, як атрибут використовується, та може приймати наступні значення:

optional — атрибут необов’язковий (значення за замовчуванням)
prohibited — атрибут заборонено використовувати
required — атрибут обов’язковий

Примітивні типи даних

Тип даних	Facets	Опис
string	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє символьний рядок.
Boolean	pattern, whiteSpace	Представляє логічне значення, яке може бути true чи false.
decimal	enumeration, pattern, totalDigits, fractionDigits, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє довільне число.
float	pattern, enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє 32-бітне число з плаваючою комою одинарної точності.
double	pattern, enumeration, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє 64-бітне число з плаваючою комою подвійної точності.
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 та часу; nH – число годин; nM – хвилин; nS - секунд.

Примітивні типи даних

Тип даних	Facets	Опис
dateTime	enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє конкретний час. Шаблон для dateTime наступний - CCYY-MM-DDThh:mm:ss, де CC - це століття; YY – рік; MM – місяць; DD – день; T - роздільник дати та часу; hh – кількість годин; mm – хвилин; ss – секунд. За потреби можна вказувати частки секунди. Наприклад, соті частки у шаблоні: ss.ss
time	enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє конкретний час дня. Шаблон для time наступний - hh:mm:ss.sss (частка мілісекунд необов'язкова).
date	enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє календарну дату. Шаблон для date наступний - CCYY-MM-DD (тут необов'язково 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).

Примітивні типи даних

Тип даних	Facets	Опис
gMonth	enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє календарний місяць(--ММ--).
hexBinary	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє довільну шістнадцятково-закодовану двійкову інформацію. HexBinary - набір двійкових октетів фіксованої довжини, який складається з чотирьох пар шістнадцяткових символів. Наприклад, 0-9a-fA-F.
base64Binary	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє довільну Base64-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 абсолютним або відносним, і може мати необов'язковий ідентифікатор фрагмента.
QName	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє складене ім'я. Ім'я складається з префіксу та локальної назви, яка відокремлена двокрапкою. І префікс, і локальні назви мають бути NCNAME. Префікс повинен бути пов'язаним з namespace URI посиланням, використовуючи оголошення простору імені.
NOTATION	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє тип атрибута СИСТЕМИ ПОЗНАЧЕНЬ. Набір QNAMES.

Похідні типи даних

Похідні типи даних	Facets	Опис
normalizedString	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє нормалізовані рядки. Цей тип даних був отриманий з string.
token	enumeration, pattern, length, minLength, maxLength, whiteSpace	Представляє маркіровані рядки. Цей тип даних був отриманий normalizedString.
language	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє ідентифікатори природної мови (певний RFC 1766). Цей тип даних був 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 та інші символи) в будь-якій комбінації. На відміну від Name та NCNAME, NMTOKEN не має ніяких обмежень на перший символ. Цей тип даних був 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 з символу, символу підкреслення або двокрапки та продовжується символами імен (символи, цифри, та інші символи). Цей тип даних був отриманий з token.

Похідні типи даних

Похідні типи даних	Facets	Опис
NCName	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє неколонковані назви. Цей тип даних - той самий, що й Name, але не може починатись з двокрапки. Цей тип даних був отриманий з Name.
ID	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє тип атрибута ID, визначений у XML 1.0 Рекомендації. ІДЕНТИФІКАТОР не повинен мати двокрапки (NCName) і повинен бути унікальним у межах XML документа. Цей тип даних був отриманий з NCNAME.
IDREF	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє посилання до елемента, що має атрибут ID, який точно відповідає встановленому ІДЕНТИФІКАТОРУ. IDREF повинен бути NCNAME і повинен бути значенням елемента або атрибута типу ID в межах XML документа. Цей тип даних був отриманий з NCNAME.
ENTITY	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє тип атрибута ENTITY. Це - посилання до об’єкту, що неможливо проаналізувати, з ім’ям, яке точно відповідає встановленому імені. ENTITY має бути NCNAME та повинен бути оголошений в схемі як неаналізоване ім’я об’єкта. Цей тип даних був 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 був отриманий з decimal.

Похідні типи даних

Похідні типи даних	Facets	Опис
nonPositiveInteger	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число, що менше або дорівнює нулю. NonPositiveInteger складається з від'ємного знаку (-) та послідовності десяткових цифр. Цей тип даних був отриманий з цілого числа.
negativeInteger	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число, що менше нуля. Цей тип даних був отриманий з nonPositiveInteger.
long	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням -9223372036854775808 та максимальним 9223372036854775807. Цей тип даних був отриманий з цілого числа.
int	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням -2147483648 та максимальним 2147483647. Цей тип даних був отриманий з long.
short	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням -32768 та максимальним 32767. Цей тип даних був отриманий з int.

Похідні типи даних

Похідні типи даних	Facets	Опис
byte	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням -128 та максимальним 127. Цей тип даних був отриманий з short.
nonNegativeInteger	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число, що більше або дорівнює нулю. Цей тип даних був отриманий з цілого числа.
unsignedLong	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням нуль та максимальним 18446744073709551615. Цей тип даних був отриманий з nonNegativeInteger.
unsignedInt	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням нуль та максимальним 4294967295. Цей тип даних був отриманий з unsignedLong.
unsignedShort	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням нуль та максимальним 65535. Цей тип даних був отриманий unsignedInt.
unsignedByte	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число з мінімальним значенням нуль та максимальним 255. Цей тип даних був отриманий з unsignedShort.

Похідні типи даних

Похідні типи даних	Facets	Опис
positiveInteger	enumeration, fractionDigits, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, totalDigits, whiteSpace	Представляє ціле число, яке більше нуля. Цей тип даних був отриманий з nonNegativeInteger.
gMonth	enumeration, pattern, minInclusive, minExclusive, maxInclusive, maxExclusive, whiteSpace	Представляє календарний місяць (--MM--).
hexBinary	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє довільну шістнадцятково-закодовану двійкову інформацію. HexBinary - набір двійкових октетів фіксованої довжини, що складається з чотирьох пар шістнадцяткових символів. Наприклад, 0-9a-fA-F.
base64Binary	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє довільну Base64-закодовану двійкову інформацію. Base64Binary - набір двійкових октетів фіксованої довжини.
anyURI	length, pattern, maxLength, minLength, enumeration, whiteSpace	Представляє URI відповідно до RFC 2396. Значення anyURI може бути абсолютним або відносним і може мати необов'язковий ідентифікатор фрагмента.
QName	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє складене ім’я. Ім'я складається з префіксу та локальної назви, що відокремлена двокрапкою. І префікс, і локальні назви мають бути NCNAME. Префікс має бути пов'язаним з namespace URI посиланням, використовуючи оголошення простору імені.
NOTATION	length, enumeration, pattern, maxLength, minLength, whiteSpace	Представляє тип атрибута СИСТЕМ ПОЗНАЧЕНЬ. Набір QNAMES.

Визначені користувачем прості типи

Обмеження використовуються для визначення допустимих значень елементів або атрибутів XML.

Визначені користувачем прості типи, отримані з вбудованих типів із застосуванням до них іменованих обмежень (**фасет**/ Facets).

Синтаксис використовування фасетів обмежень наступний:

```
<xsd:restriction base="тип_даних">
   <xsd:ім'я_фасета value="значення_фасета"/>
</xsd:restriction>
```

Визначені користувачем прості типи

Restrictions on Values (обмеження на значення)

```
<xsd:element name="age">
  <xsd:simpleType>
    <xsd:restriction base="xsd:integer">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="120"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Restrictions on a Set of Values (обмеження перерахування 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>
```

Визначені користувачем прості типи

Restrictions on a Series of Values (обмеження на групу значень - шаблон - регулярний вираз)

```
<xsd:element name="letter">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-z]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="choice">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[xyz]"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Визначені користувачем прості типи

```
<xsd:element name="letter">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="([a-z])*"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>


<xsd:element name="letter">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="([a-z][A-Z])+"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Визначені користувачем прості типи

```
<xsd:element name="gender">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="male|female"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[a-zA-Z0-9]{8}"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Визначені користувачем прості типи

Restrictions on Length (обмеження на число одиниць довжини, одиниці довжини залежать від **типу** даних**)**

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

```
<xsd:element name="password">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:minLength value="5"/>
      <xsd:maxLength value="8"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```

Визначені користувачем прості типи

Фасети обмежень простих типів

Facets	Значення
enumeration	Певний набір значень. Обмежує тип даних вказаними значеннями.
fractionDigits	Значення з певним максимальним числом десяткових цифр у дробовій частині.
length	Ціле число одиниць довжини. Одиниці довжини залежать від типу даних.
maxExclusive	Верхній поріг значень (усі значення – менші від зазначеного).
maxInclusive	Максимальне значення.
maxLength	Ціле число одиниць максимальної довжини.
minExclusive	Нижній поріг значень (усі значення – більше зазначеного).
minInclusive	Мінімальне значення.
minLength	Ціле число одиниць мінімальної довжини.
pattern	Літеральний шаблон, якому мають відповідати значення.
totalDigits	Значення з певним максимальним числом десяткових цифр.
whiteSpace	Одне з визначених значень: preserve, replace або collapse

Визначені користувачем прості типи

Значення Facet whiteSpace	Опис
preserve	Ніяка нормалізація не виконується (**НЕ** видаляються символи пробілів).
replace	Усі #x9 (tab), #xA (line feed) and #xD (carriage return) замінюються на #x20 (пробіл). Процесор XML **ЗАМІНИТЬ** усі символи пробілів (переходи рядків, табуляції, пробіли та повернення каретки) на пробіли.
collapse	Після replace-обробки всі внутрішні ланцюжки #x20 руйнуються до одного пробілу, а оточуючі пробіли прибираються. Процесор XML **ВИДАЛЯЄ** всі символи пробілів (переходи рядків, табуляції, пробіли, повернення каретки замінюються пробілами, початкові й кінцеві пробіли видаляються, а декілька пробілів зводяться до одного)

Фасеты можуть бути вказані **тільки один раз** у визначенні типу (крім **enumeration** та **pattern** (enumeration та pattern можуть мати багатократні входження і групуватись).

Якщо у схемі застосовується невідомий фасет, негайним результатом буде порушення цього обмеження, так що **простий тип**, визначений за допомогою цього фасета, **буде вилучений зі схеми, а будь-які посилання на нього розглядатимуться як невиконані посилання!**

Іменований тип даних

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

Можна створити визначення простого типу **simpleType** і назвати його, наприклад, name_type. У результаті ви отримаєте іменований тип. Після цього ви зможете застосовувати цей тип також до інших елементів у схемі. Це особливо корисно, колі у визначенні використовуються **фасети обмеження типу**, щоб не повторювати їх кожен раз в інших визначеннях.

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

Іменований тип 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" use="required"/>
<xsd:attribute name="Телефон" type="txt15pre"/>
```

whiteSpace value="preserve" - не видаляються символи пробілів

Визначення кількості екземплярів елементів

Мінімальну **частоту появи дочірнього елемента** у батьківському можна визначити за допомогою атрибута **minOccurs (**occur - зустрічатись**)**, а максимальну — за допомогою атрибута **maxOccurs**.

<xsd:element ref="note" **minOccurs**="0" **maxOccurs**="5"/>

ref="note" **посилається на інше визначення елемента** з ім'ям "note". Це означає, що детальний опис того, що може містити елемент "note", визначено в іншій частині схеми.

Значення **за замовчуванням** атрибута **minOccurs** дорівнює **1**.

Якщо значення атрибута **maxOccurs** не визначене, то його **значення за замовчуванням** дорівнює значенню атрибута **minOccurs**.

Для того щоб вказати **відсутність верхньої границі,** встановіть значення атрибута **maxOccurs, що дорівнює unbounded (**необмежений**)**.

Робота зі значеннями елементів, що задані за замовчуванням

Окрім атрибутів minOccurs і maxOccurs, для присвоєння елементу відповідного значення можна використовувати атрибути **fixed** та **default** елемента **<xsd: element>** (використовується тільки **один з цих атрибутів, а не обидва одночасно**).

Встановлення значення **fixed,** яке дорівнює **400**, означає, що значення елемента завжди повинно бути 400.

З іншого боку, встановлення значення **default**, яке дорівнює **400**, означає, що значення за замовчуванням дорівнює 400, але якщо елемент є в документі, то його фактичне значення **може бути іншим**.

Обмеження та значення атрибутів за замовчуванням

За аналогією з ЕЛЕМЕНТАМИ, можна задати тип АТРИБУТІВ, але, на відміну від ЕЛЕМЕНТІВ, **АТРИБУТИ повинні мати простий тип**. **minOccurs** та **maxOccurs** для атрибутів **не використовуються**, тому що атрибути можуть відображатись не більше ніж один раз. Для обмеження АТРИБУТІВ використовується інший синтаксис.
АТРИБУТИ оголошуються за допомогою компонента **<xsd:attribute>** при цьому <xsd:attribute> включає в себе **тип** атрибута **(простий тип)**.

Тоді, як вказати, **АТРИБУТ** обов'язковий чи ні, чи є у **АТРИБУТА** значення, що задане за замовчуванням, або чи присвоєне йому певне значення? Для цього використовують параметри **use** та **value** компонента **<xsd:attribute>.**

Обмеження та значення атрибутів за замовчуванням

Атрибут use визначає, що АТРИБУТ обов'язковий чи ні, при цьому, якщо АТРИБУТ необов'язковий, то чи встановлено значення АТРИБУТА, або воно визначається за замовчуванням. Параметр **value** містить необхідне значення.

Можливі значення параметру **use**:

- **required** — атрибут обов'язковий і може мати будь-яке значення;
- **optional** — атрибут необов'язковий і може мати будь-яке значення;
- **fixed** — значення атрибута фіксоване, і його можна встановити за допомогою параметра **value**;
- **default** — якщо атрибута нема, його значення дорівнює значенню за замовчуванням, яке встановлено для параметра **value** (якщо атрибут є в документі, то його значення дорівнює значенню, яке присвоюється йому в цьому документе);
- **prohibited** — атрибут не повинен відображатись - заборонено.

Створення простих типів

Тип catalogID є простим типом, не вбудованим у специфікацію XML-схеми. Нижче приведено порядок визначення компонента <simpleType>:

```
<xsd:simpleType name="catalogID">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-\d{4}-\d{3}">
  </xsd: restriction
</xsd:simpleType>
```

Такі нові прості типи, як **catalogID**, повинні основуватись на вже існуючому простому типі (або на вбудованому простому типі, або на вже створеному типі, тут використовується вбудований тип xsd:string). Для цього використовується параметр **base** в компоненті **<xsd:restriction>.** Розглянутий тип catalogID заснований на типі xsd:string з парою «параметр/значення» base="xsd:string".

Створення простих типів за допомогою фасетів

В DTD немає способів накласти на символьні дані **шаблон обмежень**, а в XML Schema таких способів декілька. Для опису властивостей нових простих типів XML-схеми використовують **фасети** (facets).

Використання фасетів дозволяє обмежити дані, що містяться в простому типі.

Наприклад, якщо необхідно створити простий тип dayOfMonth, який може приймати значення від 1 до 31 включно. Скористаємося двома фасетами — **minInclusive** та **maxInclusive**:

```
<xsd:simpleType name="dayOfMonth">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="1"/>
    <xsd:maxInclusive value="31"/>
  </xsd:restriction>
</xsd:simpleType>
```

Після створення нового простого типу можна визначити ЕЛЕМЕНТ і АТРИБУТИ з таким типом dayOfMonth.

Створення простих типів за допомогою фасетів

Простий тип **catalogID** використовує фасет **pattern** для **визначення регулярного виразу** (regular expression), якому мають відповідати текстові значення рядків цього типу, тобто зразок, що використовується для перевірки формату тексту:

```
<xsd:simpleType name="catalogID">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-\d{4}-\d{3}">
  </xsd:restriction>
</xsd:simpleType>
```

Текст у типі **simpleType** повинен відповідати **регулярному виразу** "\d{3}-\d{4}-\d{3}", яка означає, що текстові рядки формуються з трьох цифр, дефіса, чотирьох цифр, другого дефіса та трьох цифр.

Створення простих типів за допомогою фасетів

Фасет **enumeration** дозволяє встановлювати перелік значень аналогічно до того, як це робиться у визначеннях DTD. Використовуючи **enumeration**, можна обмежити можливі значення простого типу **списком деяких значень**.

Наприклад, простий тип weekday, що приймає значення "Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday" та "Saturday", можна визначити наступним чином:

```
<xsd:simpleType name="weekday">
<xsd:restriction base="xsd:string">
        <xsd:enumeration value="Sunday"/>
        <xsd:enumeration value="Monday"/>
        <xsd:enumeration value='Tuesday"/>
        <xsd:enumeration value="Wednesday"/>
        <xsd:enumerateon value="Thursday"/>
        <xsd:enumeration value="Friday"/>
        <xsd:enumeration value="Saturday"/>
</xsd:restriction>
</xsd:simpleType>
```

Модель контенту елемента складного типу (Complex Types)

Модель контенту елемента складного типу - формальний опис структури та допустимого вмісту елемента, яке використовується для перевірки правильності XML документа. **Моделі контенту Схеми** надують більший контроль структури елементів, ніж моделі контенту DTD. Крім того, **моделі контенту Схеми** дозволяють перевіряти правильність змішаного вмісту.

Модель контенту може обмежувати документ до деякого набору типів елементів та атрибутів, описувати й підтримувати зв'язки між цими різними компонентами та унікально позначати окремі компоненти. Вільне використання моделі контенту дозволяє розробникам змінювати структурну інформацію.

Список оголошень дочірніх елементів наводиться в структурі **групуючих** XSD-компонентів - **choice**, **sequence** та **all**.

Модель контенту елемента складного типу

Компонент **xsd:choice** дозволяє **тільки одному** з дочірніх елементів, які містяться в групі бути в складі батьківського елемента.

Компонент **xsd:sequence** потребує появи дочірніх елементів групи в точно **встановленій послідовності** в складі батьківського елемента.

Компонент **xsd:all** дозволяє дочірнім елементам в групі бути (або не бути) в будь-якій послідовності в складі батьківського елемента.

Модель контенту елемента складного типу (Complex Element)

Компонент **xsd:group** використовується для визначення групи та для посилання на іменовану групу. Ви можете використовувати **модель групи**, для визначення набору компонентів, які можуть бути повторені в документі.

Це корисно для формування визначення **комплексного типу (complexType)**.

Іменовану модель групи можна далі визначити, використовуючи <**xsd:sequence**>, <**xsd:choice**> або <**xsd:all**>.

Іменовані групи повинні визначатись в **корені Схеми**. За необхідності багаторазового використання списку елементів, який визначено в групі, не треба кожен раз писати цей список – достатньо дати посилання на **іменовану групу**

```
<xsd:group ref="ім’я_групи">
```

Complex Types

Модель контенту елемента складного типу (Complex Element) — це елемент XML, який може містити **інші елементи** та/або **атрибути** та будь-які допустимі **фасети**.

Зазвичай, складні типи містять набір оголошень **елементів**, набір оголошень **атрибутів** та **елементних посилань**.

```
<xsd:element name="i’мя_элемента" type="xsd:тип_даних">
  <xsd:complexType>
    <xsd:sequence>
       <xsd:element name="i’мя_элемента1" type="xsd:тип_даних "/>
    </xsd:sequence>
    <xsd:attribute name="i’мя_атрибута" type="xsd:тип_даних "/>
  </xsd:complexType>
</xsd:element>
```

Complex Types

Можна створювати складний елемент двома способами:

1) Спочатку визначити складний тип, потім створити елемент, атрибут **type** якого посилається на ім'я цього складного типу

```
<xsd:complexType name = "StudentType">
  <xsd:sequence>
    <xsd:element name = "firstname" type = "xsd:string "use="required"/>
    <xsd:element name = "lastname" type = "xsd:string" use="required"/>
    <xsd:element name = "nickname" type = "xsd:string“ use="optional"/>
    <xsd:element name = "marks" type = "xsd:positiveInteger" use="required"/>
  </xsd:sequence>
  <xsd:attribute name = "student_id" type = "xsd:positiveInteger" use="required" />
</xsd:complexType>

<xsd:element name = "student" type = "StudentType"/>
```

Complex Types

2) Визначити складний тип всередині декларації елемента.

```
<xsd:element name = "student">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name = "firstname" type = "xsd:string"/>
      <xsd:element name = "lastname" type = "xsd:string"/>
      <xsd:element name = "nickname" type = "xsd:string"/>
      <xsd:element name = "marks" type = "xsd:positiveInteger"/>
    </xsd:sequence>
  <xsd:attribute name = "student_id" type = 'xsd:positiveInteger'/>
  </xsd:complexType>
<xsd:element>
```

Complex Types

```
<xsd:complexType name="personinfo">
  <xsd:sequence>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="fullpersoninfo">
  <xsd:complexContent>
    <xsd:extension base="personinfo">
      <xsd:sequence>
        <xsd:element name="address" type="xsd:string"/>
        <xsd:element name="city" type="xsd:string"/>
        <xsd:element name="country" type="xsd:string"/>
      </xsd:sequence>
    </xsd:extension>
  </xsd:complexContent>
</xsd:complexType>

<xsd:element name="employee" type="fullpersoninfo"/>
```

Complex Empty Elements

Complex Empty Elements

Порожній складний елемент не може мати контенту, **тільки атрибути**. Щоб визначити тип без контенту, ми повинні визначити тип, який дозволяє використовувати елементи в своєму вмісті, але насправді ми не оголошуємо ніяких елементів, наприклад:

```
<xsd:complexType name="prodtype">
  <xsd:attribute name="prodid" type="xsd:positiveInteger"/>
</xsd:complexType>

<xsd:element name="product" type="prodtype"/>
```

Порожній елемент XML:

```
<product prodid="1345" />
```

Complex Empty Elements

Complex Empty Elements

```
<xsd:element name="product">
  <xsd:complexType>
    <xsd:complexContent>
      <xsd:restriction base="xsd:integer">
        <xsd:attribute name="prodid" type="xsd:positiveInteger"/>
      </xsd:restriction>
    </xsd:complexContent>
  </xsd:complexType>
</xsd:element>
```

Компонент **complexContent** сигналізує про те, що ми маємо намір обмежити або розширити **модель вмісту складного типу**, а обмеження **integer** оголошує один атрибут, але не вводить ніякого вмісту елементу.

Complex Types Containing Elements Only

Complex Types Containing Elements Only

Складні типи, що містять тільки елементи

```
<xsd:element name="person">
  <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:sequence - означає, що визначені елементи (**firstname** та **lastname**) мають з’являтись в цьому порядку всередині елемента **person.**

Complex Text-Only Elements

Складні текстові елементи (**Complex Text-Only Elements)** - це тип елемента, який може містити текстові дані, але не інші дочірні елементи. Він визначається за допомогою **complexType** з обмеженням **simpleContent**.

Цей тип містить **тільки простий вміст** (текст і атрибути). При використанні простого контенту ви повинні визначити розширення або обмеження в елементі **simpleContent**, наприклад:

```
<xsd:element name="somename">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="base_type">
        ....
        ....
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Complex Text-Only Elements

Complex Text-Only Elements

```
<xsd:element name="somename">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:restriction base="base_type">
        ....
        ....
      </xsd:restriction>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

Використовуйте елемент розширення / обмеження, щоб розширити чи обмежити базовий простий тип (**simple type**) елемента.

Complex Types with Mixed Content

Можна створювати елементи, які підтримують **змішаний вміст** (**mixed content**): **текст** та інші **елементи**. Елементи зі змішаним контентом можна створювати за допомогою, як схем, так і визначень DTD. В цих елементах **символьні дані** можуть з'являтись на **тому ж рівні**, що й дочірні елементи.

```
<xsd:complexType name="lettertype" mixed="true">
  <xsd:sequence>
    <xsd:element name="name" type="xsd:string"/>
    <xsd:element name="orderid" type="xsd:positiveInteger"/>
    <xsd:element name="shipdate" type="xsd:date"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:element name="letter" type="lettertype"/>
```

Complex Types with Mixed Content

Complex Types with Mixed Content

Елемент змішаного складного типу може містити **атрибути**, **елементи** і **текст**.

```
<xsd:element name="letter">
  <xsd:complexType mixed="true">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="orderid" type="xsd:positiveInteger"/>
      <xsd:element name="shipdate" type="xsd:date"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

.xsd

Для того, щоб у файлі XML символьні дані відображались між дочірніми елементами для батьківського «letter», у атрибуті **mixed** повинно бути встановлено значення "true"

```
<letter>
  Dear Mr. <name>John Smith</name>.
  Your order <orderid>1032</orderid>
  will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

.xml

Створення елементів зі змішаним вмістом

В DTD **неможна** задати порядок та кількість дочірніх елементів, які з'являються в елементі з **типом змішаного вмісту**. Проте **Схеми** забезпечують додаткові можливості, коли **порядок** та **кількість дочірніх елементів** відповідає заданим у схемі показниками.

Визначення DTD дають тільки часткову специфікацію синтаксису типів зі змішаним вмістом, **Схеми** передбачають більш повну специфікацію синтаксису.

Indicators

Ми можемо контролювати за допомогою індикаторів, як елементи повинні використовуватись в документах.

Order indicators (індикатори **порядку появи елемента):**

- All
- Choice
- Sequence

Occurrence indicators (індикатори **числа поява елемента):**

- maxOccurs
- minOccurs

Group indicators (індикатори **груп):**

- Group name
- attributeGroup name

Order Indicators

Індикатори порядку використовуються для визначення порядку появи елементів.

Індикатор порядку <all>

Індикатор <**all**> вказує на те, що дочірні елементи можуть з'являтись в **будь-якому** порядку в батьківському елементі і що кожен дочірній елемент повинен зустрічатись **лише один раз**:

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:all>
      <xsd:element name="firstname" type="xsd:string"/>
      <xsd:element name="lastname" type="xsd:string"/>
    </xsd:all>
  </xsd:complexType>
</xsd:element>
```

Order Indicators

Індикатор порядку <choice> вказує на те, що може зустрічатись **або** один дочірній елемент, **або** інший в батьківському елементі:

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:choice>
      <xsd:element name="employee" type="employee"/>
      <xsd:element name="member" type="member"/>
    </xsd:choice>
  </xsd:complexType>
</xsd:element>
```

Order Indicators

Індикатор порядку <**sequence**> вказує на те, що дочірні елементи **повинні з'являтись в певному порядку** у батьківському елементі:

```
<xsd:element name="person">
  <xsd:complexType>
   <xsd:sequence>
     <xsd:element name="firstname" type="xsd:string"/>
     <xsd:element name="lastname" type="xsd:string"/>
   </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

Occurrence Indicators

Індикатори входження використовуються для визначення того, скільки разів дочірній елемент може з'являтися у батьківському елементі.

Для всіх індикаторів (any, all, choice, sequence, group name, group reference) **значення за замовчуванням** для maxOccurs і minOccurs дорівнює **1**.

Occurrence Indicators

Індикатор <maxOccurs> встановлює **максимальну** кількість разів появи дочірнього елемента в батьківському елементі:

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="full_name" type="xsd:string"/>
      <xsd:element name="child_name" type="xsd:string" maxOccurs="10"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Occurrence Indicators

Індикатор maxOccurs

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="full_name" type="xsd:string"/>
      <xsd:element name="child_name" type="xsd:string"
      maxOccurs="unbounded" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Для того, щоб елемент міг з’являтись **необмежену** кількість разів, використовуйте **maxOccurs** = "**unbounded**"

Occurrence Indicators

Індикатор <minOccurs> встановлює мінімальну кількість разів появи елемента:

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="full_name" type="xsd:string"/>
      <xsd:element name="child_name" type="xsd:string" maxOccurs="10"
        minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Group Indicators

Індикатори групи використовуються для визначення пов’язаних наборів елементів.

Групи елементів

Групи елементів визначаються за допомогою оголошення групи, наприклад:

```
<xsd:group name="groupname">
...
</xsd:group>
```

Ви повинні визначити елемент **all, choice** або **sequence** всередині оголошення групи.

Group Indicators

Групи елементів

В наступному прикладі описується група з ім'ям «persongroup», що визначає групу елементів, які мають знаходитись в точній послідовності:

```
<xsd:group name="persongroup">
  <xsd:sequence>
    <xsd:element name="firstname" type="xsd:string"/>
    <xsd:element name="lastname" type="xsd:string"/>
    <xsd:element name="birthday" type="xsd:date"/>
  </xsd:sequence>
</xsd:group>
```

Group Indicators

Групи атрибутів

Групи атрибутів визначаються за допомогою оголошення **attributeGroup**:

```
<xsd:attributeGroup name="groupname">
...
</xsd:attributeGroup>
```

Group Indicators

Групи атрибутів

Після того, як ви визначили групу атрибутів, ви можете посилатись на неї в іншому визначенні, наприклад:

```
<xsd:attributeGroup name="personattrgroup">
  <xsd:attribute name="firstname" type="xsd:string"/>
  <xsd:attribute name="lastname" type="xsd:string"/>
  <xsd:attribute name="birthday" type="xsd:date"/>
</xsd:attributeGroup>

<xsd:element name="person">
  <xsd:complexType>
    <xsd:attributeGroup ref="personattrgroup"/>
  </xsd:complexType>
</xsd:element>
```

Компонент <any>

Компонент <**any**> дозволяє нам розширювати XML-документ **елементами, не вказаними у схемі!**

Наступний приклад - це фрагмент XML-схеми. Він демонструє оголошення для елементу "person". Використовуючи компонент <**any**>, ми можемо **розширювати** контент "person" будь-яким елементом (після <lastname>) :

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="firstname" type="xsd:string"/>
      <xsd:element name="lastname" type="xsd:string"/>
      <xsd:any minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Компонент <anyAttribute>

Компоненти <**any**> та <**anyAttribute**> використовуються для створення розширюваних документів! Вони дозволяють документам містити **додаткові** елементи, які не були оголошені в схемі XML.

Використовуючи компонент anyAttribute, ми можемо додавати будь-яку кількість атрибутів до елемента "person" у схемі family.xsd:

```
<xsd:element name="person">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="firstname" type="xsd:string"/>
      <xsd:element name="lastname" type="xsd:string"/>
    </xsd:sequence>
    <xsd:anyAttribute/>
  </xsd:complexType>
</xsd:element>
```

family.xsd

Компонент <anyAttribute>

Тепер нам необхідно розширити елемент «person» атрибутом «eyecolor» у схемі **family.xsd**. В цьому випадку ми можемо це зробити, навіть якщо автор схеми вище ніколи не оголошував атрибут «колір очей»:

```
<xsd:attribute name="eyecolor">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="blue|brown|green|grey"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:attribute>
```

attribute.xsd

Елемент <anyAttribute>

В XML-файлі нижче (що має назву «Myfamily.xml») використовуються компоненти з двох різних схем; "family.xsd" і "attribute.xsd":

```
<?xml version="1.0" encoding="UTF-8"?>
<persons xmlns="http://www.microsoft.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:SchemaLocation="http://www.microsoft.com family.xsd
https://www.w3schools.com attribute.xsd">
  <person eyecolor="green">
    <firstname>Hege</firstname>
    <lastname>Refsnes</lastname>
  </person>
  <person eyecolor="blue">
    <firstname>Stale</firstname>
    <lastname>Refsnes</lastname>
  </person>
</persons>
```

Myfamily.xml

Переваги XSD

Одна з найсильніших сторін **Схем XML** - це **підтримка типів даних:**
Легше описати допустимий вміст документа
Легше перевірити правильність даних
Легше визначати фасети даних (обмеження даних)
Легше визначати шаблони даних (формати даних)
Легше конвертувати дані між різними типами даних.

Xml-Схеми використовують синтаксис XML:
Ще одна сильна сторона XML-схем полягає в тому, що вони написані на XML.
Ви можете використовувати свій XML-парсер для аналізу файлів Схеми.
Ви можете керувати своєю Схемою за допомогою **XML DOM**
Ви можете перетворювати свою Схему за допомогою **XSLT**

Переваги XSD

XML-схеми можна розширювати, оскільки вони написані на XML:

- Повторно використовуйте вашу Схему в інших схемах.
- Створюйте власні типи даних, похідні від стандартних типів.
- Можна посилатись на кілька Схем в одному документі

XML-схеми - безпечна передача даних

Більш розвинені можливості

Оголошення **можна групувати для спадкування** однакових властивостей та забезпечення моделювання більш складного вмісту, крім того, вони можуть успадковувати властивості інших оголошень (в об'єктно-орієнтованому стилі).

XML Schema надає альтернативу DTD, **але це зовсім не заміна DTD**.

DTD зберігають за собою низку переваг: компактний розмір, знайомий синтаксис, простоту. **Разом вони забезпечують різними методами досягнення подібних цілей.**

Література

On-line підручники:

https://coderlessons.com/tutorials/xml-tekhnologii/uchitsia-xsd/uchebnik-po-xsd

https://www.tutorialspoint.com/xml/xml_schemas.htm