День 1

593. Разработка схем XML-документов с использованием XML Schema

лектор

Арбатский Евгений Викторович

Состав курса

Примерная структура курса

1. DTD декларация XML документов.
2. XML схемы.
3. Стратегии проектирования XML схем.
4. RELAX NG
5. Применение XML схем.
6. Проверка документов на Java.

Распределение по дням

День 1	День 2
1. DTD декларация XML документов. 2. XML схемы.	3. Стратегии проектирования XML схем. 4. RELAX NG 5. Применение XML схем. 6. Проверка документов на Java.

Распорядок дня

10:00-13:00 — Первый блок

13:00-14:00 — Обед

14:00-16:00 — Практика, второй блок

XML

eXtensible Markup Language

расширяемый язык разметки

Правильность XML документов

Стандартом определены два уровня правильности документа XML:

- Правильно построенный (Well-formed);
- Действительный (Valid):
  - Допустимый по типу (type-valid XML document)
  - Недопустимый по типу (not-type-valid XML document)
  - Допустимый по схеме
  - Недопустимый по схеме

DTD

Document Type Definition

- **Язык описания структуры документа**
- **Описывает:**
  - Какие элементы могут присутствовать в документе
  - Повторения элементов
  - Какие атрибуты могут быть у элементов
  - Какие атрибуты обязательны
  - Какие сущности могут применятся

DTD

Способы программного анализа документа

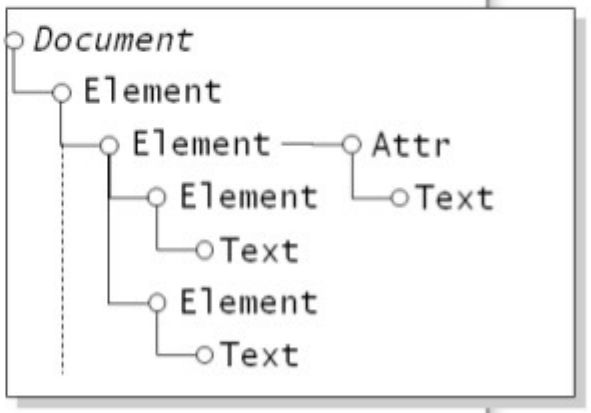
- Для программного анализа испол XML
- Парсер – синтаксический анализа
- SAX – Simple API of XML
- DOM – document object model

DOM структура

- Парсер представляет документ как иреархию объектов
- Объекты DOM – это узлы (node) связанные друг с другом

Объект Document – основной объект документа

Другие объекты представляют элементы, текст, атрибуты, комментарии и т.д.



XML

Виды DTD

- Встроенный (inline)
- Внешний (external)

external

```
<?xml version="1.0"?>
<!DOCTYPE weather-report SYSTEM "weather.dtd">
```

Inline DTD

```
<?xml version="1.0"?>
<!DOCTYPE weather-report
[
<!ELEMENT weather-report
          (date,time,area,measurements)>
<!ELEMENT date        (#PCDATA)>    ....
]>
<weather-report>
  <date>   ....
```

Пример DTD

```
<!ELEMENT pricelist (book+)>
<!ELEMENT book (title, author+, price,
  description?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT description (#PCDATA)>
<!ATTLIST price currency CDATA #IMPLIED>
```

Описание элементов

- ELEMENT – определение элемента
- ATTLIST – определение атрибута
- ENTITY – определение сущности

- Модификаторы
- * – ноль или много
- ? – ноль или один
- + – один или много

EMPTY	no content
ANY	no constraints on content
|	choice list
,	sequence
Cardinality	
	exactly one
+	one or more
?	zero or one
*	zero or more
()	grouping
(#PCDATA)	characters
(#PCDATA | ...) *	characters or elements ("mixed content")

Подключение DTD

```
<?xml version=1.0" encoding="UTF-8"?>
<!DOCTYPE DOCUMENT [
  <!ELEMENT DOCUMENT (GREETING, ME
  <!ELEMENT GREETING (#PCDATA)>
  <!ELEMENT MESSAGE (#PCDATA)>
]>

<DOCUMENT>
  <GREETING> Hello From XML </GREETING>
        <MESSAGE>
                        Welcome to the wild
                        and woolly world of
  </MESSAGE>
</DOCUMENT>
```

```
<?xml version = "1.0" standalone="no"?>

<!DOCTYPE DOCUMENT SYSTEM "mydoc.dtd">

<DOCUMENT>
  <CUSTOMER>
  ...
  </CUSTOMER>
</DOCUMENT>
```

XML Schema

XML Schema используется для определения содержимого и структуры документов. В чем-то она схожа с DTD. Но есть усовершенствования: она позволяет определить наборы символов и их взаимосвязи, обязательность и необязательность значение, множественность.

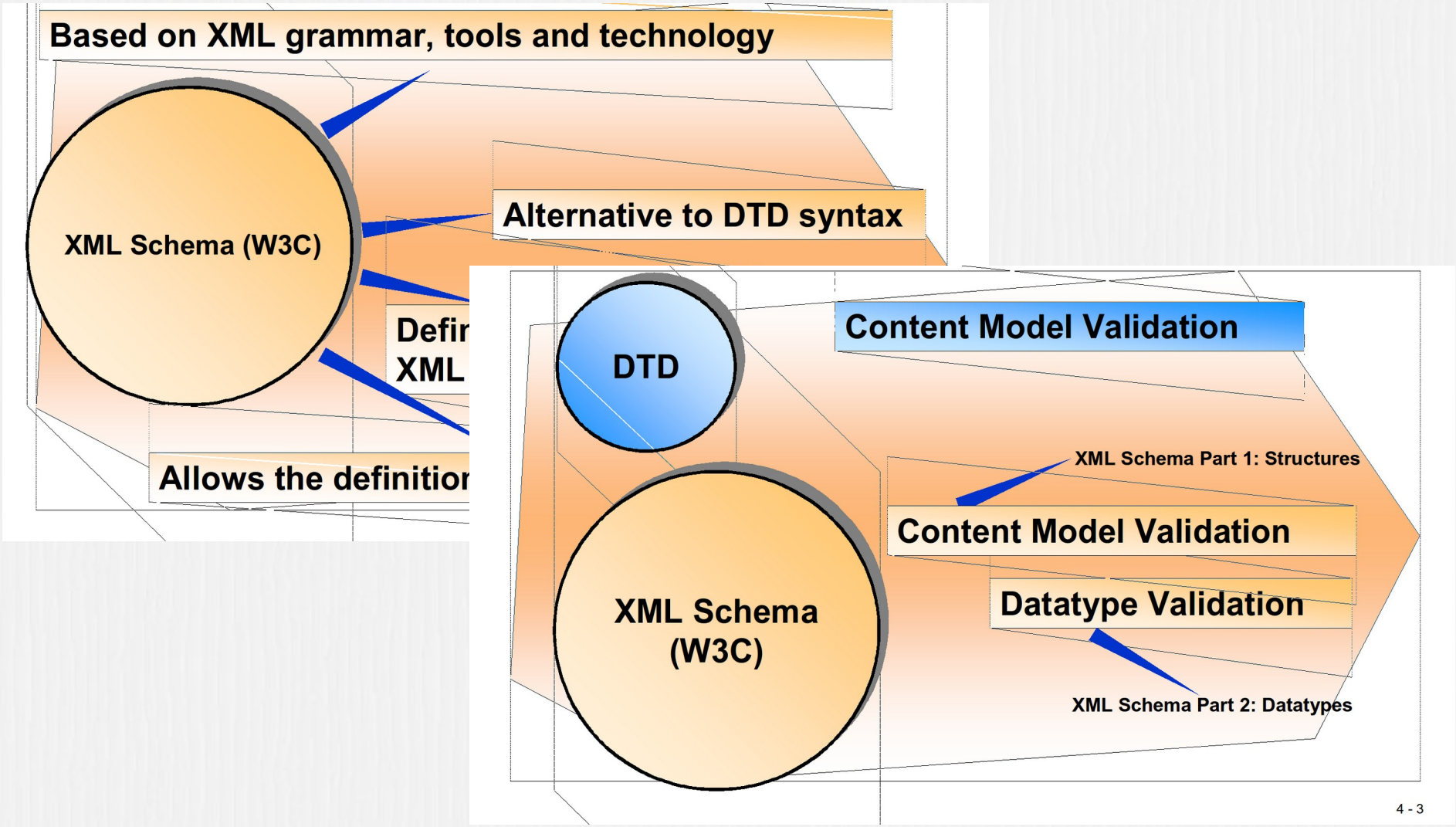
XML-документы делятся на совместимые по схеме и несовместимые по схеме. Несовместимый по схеме XML-документ при этом может оставаться совместимым по типу.

XML Schema

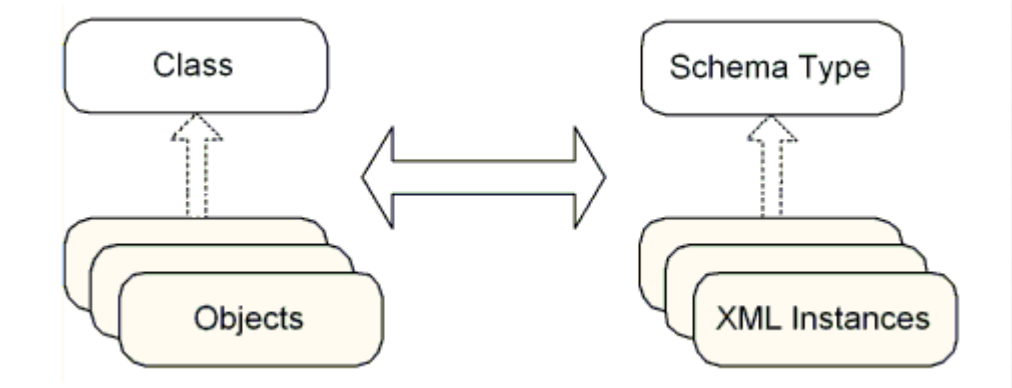
Поддержка типов данных обеспечивает:

- более легкое описание возможное содержание документа;
- более легкую проверку корректности данных;
- более легкую работу с данными из базы данных;
- более легкое определение области значений данных (ограничения данных);
- более легкое определение шаблонов данных (форматов данных);
- более легкое преобразование данных между различными типами данных.

XML Schema


Based on XML grammar, tools and technology
XML Schema (W3C)
Alternative to DTD syntax
Defir
XML
Allows the definitior
DTD
Content Model Validation
XML Schema Part 1: Structures
Content Model Validation
Datatype Validation
XML Schema Part 2: Datatypes
XML Schema (W3C)
4 - 3

XML Schema и ООП



XML Schema

Когда данные посылаются от отправителя к получателю очень важно то, что они имеют одинаковое представление о содержимом. Посредством XML Схем отправитель может описать данные таким образом, что получатель будет понимать их.

Например, дата представленная как 1999-03-11 может в ряде стран интерпретироваться как 3 ноября, в других странах как 11 марта. Однако, XML element с типом данных: <date type="date">1999-03-11</date> позволит однозначно понимать содержимое, т.к. тип данных XML даты описывается в формате CCYY-MM-DD.

XML Schema

Простой пример XML Schema, расположенный в файле "country.xsd", описывает данные о стране:

```
<xs:schema
 xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="страна" type="страна"/>
 <xs:complexType name="страна">
  <xs:sequence>
   <xs:element name="название" type="xs:string"/>
   <xs:element name="население" type="xs:decimal"/>
  </xs:sequence>
 </xs:complexType>
</xs:schema>
```

Пример документа, отвечающего этой схеме:

```
<страна
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="country.xsd">
  <название>Франция</название>
  <население>59.7</население>
</страна>
```

Включение схемы

Если не определено пространство имен

```
<building xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="task05.xsd">
```

Если есть пространство имен

```
<note
xmlns="http://www.w3schools.com"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.w3schools.com note.xsd">
```

Схема

```
<?xml version="1.0"?>
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com">
...
</xs:schema>
```

Элемент xs:schema *определяет то, что находится в пространстве имен*, а атрибут targetNamespace *определяет имя пространства имен*.

Схема

```
<?xml version="1.0"?>

<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.w3schools.com"
xmlns="http://www.w3schools.com">
...
...
</xs:schema>
```

Простой элемент

<xs:element name="xxx" type="yyy"/>

```
<lastname>Refsnes</lastname>
<age>36</age>
<dateborn>1970-03-27</dateborn>

<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Задание 1

Документ и схема без пространства имен

1. Создать пустой XML документ с корневым элементом building
2. Создать пустую XML схему
3. Подключить XML схему к XML документу
4. Проверить валидность XML документа по схеме

xmllint --schema s01.xsd s01.xml

использовать **element** с типом **string**

Задание 2

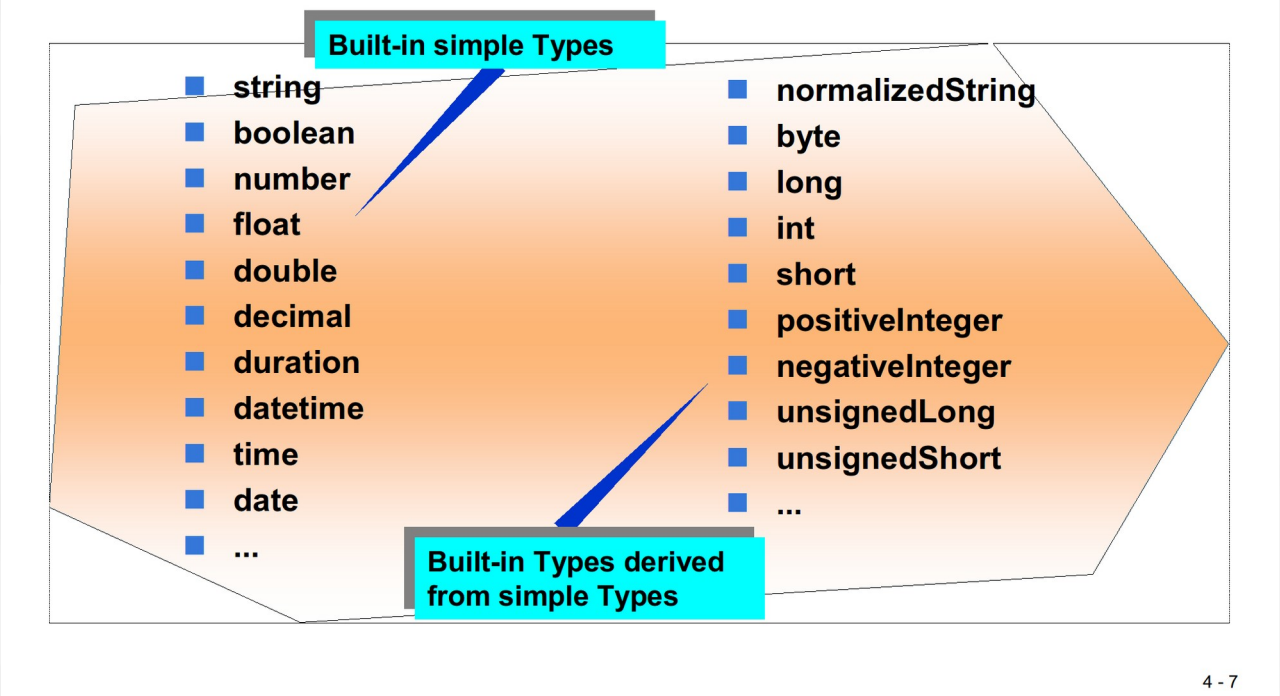
Документ и схема с пространством имен study

1. Создать пустой XML документ с корневым элементом building
2. Создать пустую XML схему
3. Подключить XML схему к XML документу
4. Проверить валидность XML документа по схеме

Значения

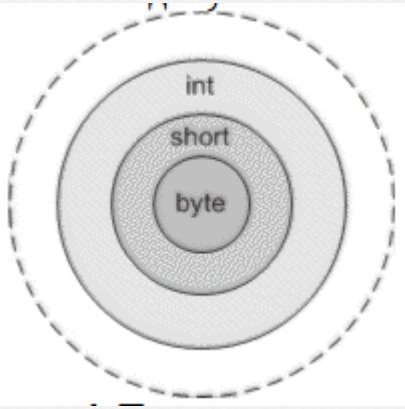
```
<xs:element name="color" type="xs:string" default="red"/>

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

Типы данных


Built-in simple Types
string
boolean
number
float
double
decimal
duration
datetime
time
date
...
normalizedString
byte
long
int
short
positiveInteger
negativeInteger
unsignedLong
unsignedShort
...
Built-in Types derived from simple Types
4 - 7

Типы данных

XML Schema предоставляет набор встроенных типов данных, которые могут использоваться разработчиками для включения текста. Все эти типы помещены в пространство имен http://www.w3.org/2001/XMLSchema. Многие встроенные типы определяются как подмножество области значений другого типа. Это явление также известно как наследование по ограничению (derivation by restriction). Например, область значений byte является подмножеством области значений short, которая является подмножеством области значений int, которая, в свою очередь, есть подмножество области значений long и т.д.



Типы данных

Normalized string — текст, из которого исключаются символы перевода строки, табуляции.

Token string — текст, из которого исключаются символы перевода строки, табуляции, пробелы по краям, множественные пробелы.

Date — дата в формате YYYY-MM-DD

Time — время в формате HH:MM:SS[+-offset]

DateTime — YYYY-MM-DD**T**HH:MM:SS

Типы данных

Period — временной интервал, который записывается в формате "PnYnMnDTnHnMnS", где:

- P — индикатор периода (**необходимо**)
- nY — количество лет
- nM — количество месяцев
- nD — количество дней
- T — начало секции со временем (требуется при наличии времени)
- nH — количество часов
- nM — количество минут
- nS — количество секунд

<period>PT15H</period>
<period>-P10D</period>

Атрибуты

<xs:attribute name="xxx" type="yyy"/>

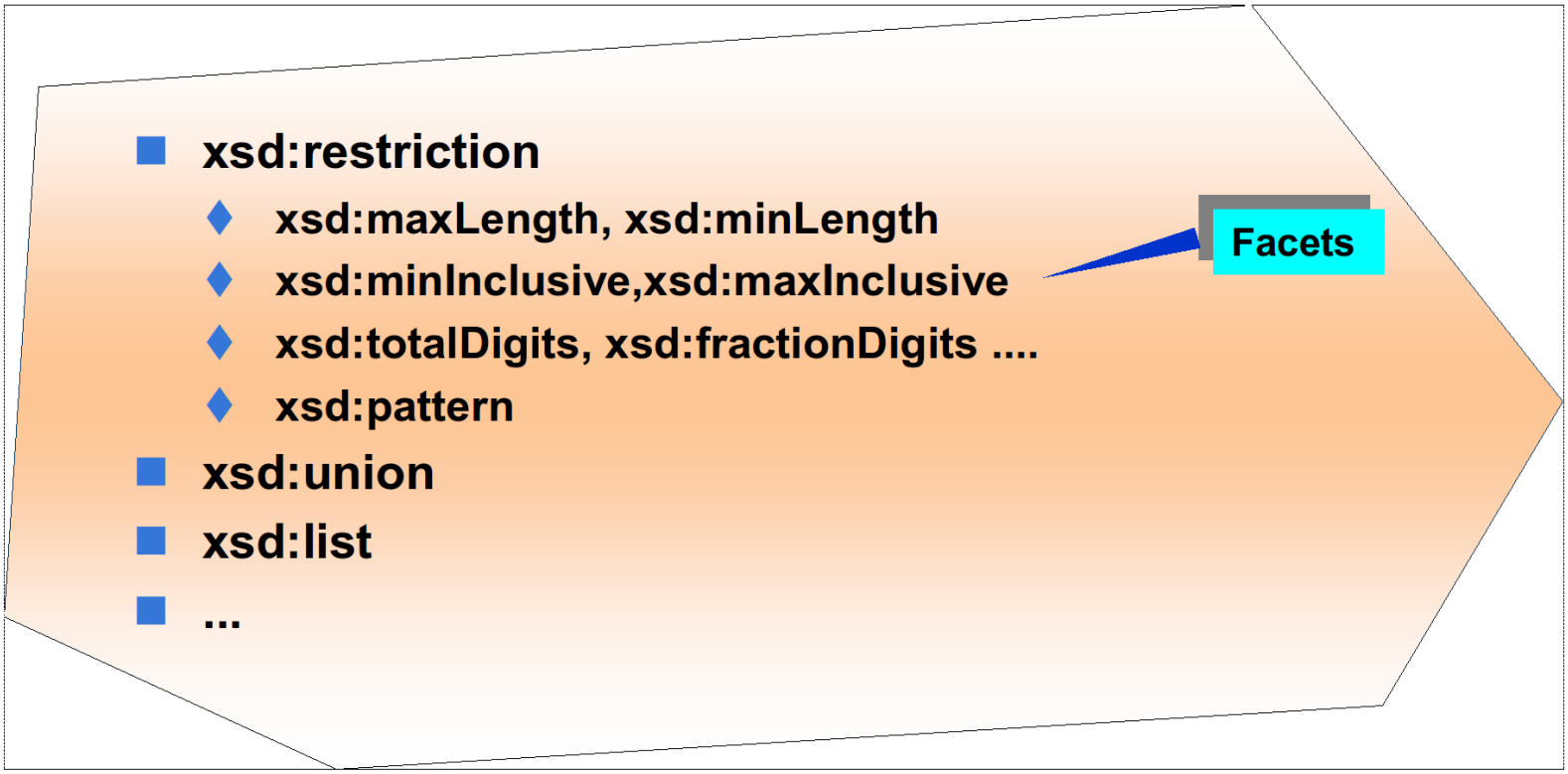
```
<lastname lang="EN">Smith</lastname>

<xs:attribute name="lang" type="xs:string"/>

<xs:attribute name="lang" type="xs:string" default="EN"/>
<xs:attribute name="lang" type="xs:string" fixed="EN"/>

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

Определение простых типов

Большинство языков программирования позволяют разработчикам только компоновать различные встроенные типы в некоторого рода структурированные типы, но они не дают возможности определять новые простые типы, имеющие определенные пользователем области значений. В XML Schema пользователям позволяется определять собственные специальные *простые типы*, области значений которых являются подмножествами предопределенных встроенных типов.

Вы определяете новый простой тип, используя элемент **xs:simpleType**. В элементе xs:simpleType вы определяете базовый тип, чью область значений вы хотите ограничить (используя элемент xs:restriction). В элементе xs:restriction вы определяете, как именно вы хотите ограничивать базовый тип, сужая один или более из его аспектов (facets).

Ограничения для простых типов



Ограничения для простых типов

Производный элемент	Описание
xsd:restriction	Новый тип является ограничением существующего типа, т.е. он имеет более узкий ряд допустимых значений.
xsd:list	Новый тип является разделенным пробелами списком другого простого типа.
xsd:union	Новый тип является объединением двух или более других простых типов.

Ограничения для простых типов restriction

Facet элемент	Описание
xsd:enumeration	Определяет фиксированное значение, с которым должен совпадать тип.
xsd:fractionDigits	Определяет максимальное количество десятичных знаков справа от десятичной точки.
xsd:length	Определяет количество символов в строковом типе, количество байтов в двоичном типе или количество элементов в списочном типе.
xsd:maxExclusive	Определяет исключающую верхнюю границу области значений типа.
xsd:maxInclusive	Определяет включающую верхнюю границу области значений типа.
xsd:maxLength	Определяет максимальное количество символов в строковом типе, максимальное количество байтов в двоичном типе или максимальное количество элементов в списочном типе.
xsd:minExclusive	Определяет исключающую нижнюю границу области значений типа.
xsd:minInclusive	Определяет включающую нижнюю границу области значений типа.
xsd:minLength	Определяет минимальное количество символов в строковом типе, минимальное количество байтов в двоичном типе или минимальное количество элементов в списочном типе.
xsd:pattern	Определяет шаблон, основанный на регулярном выражении, с которым должен совпадать тип.
xsd:totalDigits	Определяет максимальное количество десятичных знаков для типов, унаследованных от number.
xsd:whiteSpace	Определяет правила нормирования пробелов.

Ограничения enumeration

```
<xs:element name="car">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Audi"/>
      <xs:enumeration value="Golf"/>
      <xs:enumeration value="BMW"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Ограничения minInclusive, maxInclusive

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

Ограничения pattern

```
<xs:element name="initials">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Задание 3

- Определить простой текстовый тип building, который должен позволять содержать значения только из 5 латинских символов.
- Проверить на элементе вида: <building>ABCDE</building>
- Проверить на элементе вида: <building>АБВГД</building>
- Проверить на элементе вида: <building>12</building>

Анонимные типы

При создании схем применяется два стиля. Схемы могут создаваться путем определения поименованных типов (например, PurchaseOrderType) с последующим объявлением элементов этого типа (например, purchaseOrder). При этом объявленные элементы ссылаются на поименованный тип с помощью конструкции type=

Анонимный тип нет необходимости именовать и, следовательно, задавать на него ссылки. Наличие в схеме анонимного типа можно идентифицировать за счет отсутствия в объявлении элементов или атрибутов параметра type=, и присутствия непоименованного определения простого или комплексного типа.

Анонимные типы

```
<xsd:complexType name="Items">
 <xsd:sequence>
  <xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
   <xsd:complexType>
    <xsd:sequence>
     <xsd:element name="productName" type="xsd:string"/>
     <xsd:element name="quantity">
      <xsd:simpleType>
       <xsd:restriction base="xsd:positiveInteger">
        <xsd:maxExclusive value="100"/>
       </xsd:restriction>
      </xsd:simpleType>
     </xsd:element>
     <xsd:element name="USPrice"  type="xsd:decimal"/>
     <xsd:element ref="comment"   minOccurs="0"/>
     <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
    </xsd:sequence>
    <xsd:attribute name="partNum" type="SKU" use="required"/>
   </xsd:complexType>
  </xsd:element>
 </xsd:sequence>
</xsd:complexType>
```

Задание 4

- Определить простой числовой тип passnum, который должен позволять содержать значения номера паспорта в формате XXXXXX. Минимальное значение 100100.
- Проверить на элементе вида: <passnum>10000</passnum>
- Проверить на элементе вида: <passnum>100000</passnum>
- Проверить на элементе вида: <passnum>123456</passnum>

Ограничение list

Используя xsd:list, вы, по существу, определяете разделенный пробелами список значений из определенной области значений. Не стоит и упоминать о том, что при использовании xsd:list или xsd:union нет иерархии наследования, поэтому в этих случаях не выполняется совместимость типов. В следующем примере новый тип AuthorList определяется как список SSN значений.

```
<xsd:simpleType name="AuthorList">
  <xsd:list itemType="tns:SSN"/>
</xsd:simpleType>
<xsd:element name="authors" type="tns:AuthorList"/>
```

следующий документ содержит корректный экземпляр элемента authors:

```
<x:authors xmlns:x="http://example.org/publishing"> 111-11-1111 222-22-2222 333-33-3333 444-44-4444</x:authors>
```

Ограничение union

В случае использования xsd:union вы создаете новый тип, комбинирующий множество областей значений в новую область значений. Экземпляр типа-объединения может быть значением из любой из указанных областей значений. Например, следующий тип AuthorId сочетает SSN область значений с областью значений PublisherAssignedId:

```
<xsd:simpleType name="AuthorId">
<xsd:union memberTypes="tns:SSN tns:PublisherAssignedId"/>
</xsd:simpleType>
<xsd:element name="authorId" type="tns:AuthorId"/>
```

Каждый из следующих документов демонстрирует корректный экземпляр элемента authorId:

```
<x:authorId xmlns:x="http://example.org/publishing"
>111-11-1111</x:authorId>
<x:authorId xmlns:x="http://example.org/publishing"
>22-22222222</x:authorId>
```

Задание 5

- Определить простой тип passnum, который должен позволять содержать значения номера паспорта в формате XXXX XXXXXX или XXAA XXXXXX.
- Проверить на элементе вида: <passnum>1200 121212</passnum>
- Проверить на элементе вида: <passnum>12CT 121212</passnum>
- Проверить на элементе вида: <passnum>123456</passnum>

Сложные типы

XML Schema делает возможным компоновать различные простые типы (или области значений) в структуру, также известную как *составной/сложный (complex) тип*. Чтобы определить новый сложный тип в целевом пространстве имен схемы, вы используете элемент xsd:complexType:

```
...
  <xsd:complexType name="AuthorType">
    <!-- compositor goes here -->
  </xsd:complexType>
```

Сложные типы

```
<employee>
  <firstname>John</firstname>
  <lastname>Smith</lastname>
</employee>

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

Сложные типы

Элемент xsd:complexType содержит то, что известно как составитель, который описывает формирование содержимого типа, также известного как модель содержимого. XML Schema определяет три составителя, которые могут использоваться при описании составного типа, включая xsd:sequence, xsd:choice и xsd:all.

Составители содержат части, которые включают такие вещи, как другие составители, описания элементов, групповые символы и типовые группы. *Описания атрибутов не считаются частями, потому что они не повторяются. Следовательно, описания атрибутов помещаются не в составителе, а после него в конце определения составного типа.*

Составитель	Описание
xsd:sequence	Упорядоченная последовательность содержащихся частей
xsd:choice	Выбор содержащихся частей
xsd:all	Все содержащиеся части в любом порядке

XSD индикаторы

```
<xs:element name="person">
  <xs:complexType>
    <xs:all> - любой порядок для "детей", но каждый может появиться только раз
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:all>
  </xs:complexType>
</xs:element>
```

XSD индикаторы

```
<xs:element name="person">
  <xs:complexType>
    <xs:choice> - может появиться какой-то один из "детей"
      <xs:element name="employee" type="employee"/>
      <xs:element name="member" type="member"/>
    </xs:choice>
  </xs:complexType>
</xs:element>
```

XSD индикаторы

```
<xs:element name="person">
   <xs:complexType>
    <xs:sequence> - определенная последовательность "детей"
      <xs:element name="firstname" type="xs:string"/>
      <xs:element name="lastname" type="xs:string"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

XSD индикаторы

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

XSD индикаторы

Обратите также внимание, что с помощью атрибутов minOccurs и maxOccurs описание элемента определяет ограничения появлений (occurrence constraints). В составном типе ограничения появлений могут применяться к любой части. Стандартное значение для каждой части – 1, это означает, что данная часть должна появиться в указанном месте именно один раз. Определение minOccurs="0" делает данную часть необязательной, а определение maxOccurs="unbounded" позволяет бесконечные повторения части. Если пожелаете, вы можете задать также произвольные границы, например, minOccurs="3" maxOccurs="77".

Задание 6

- Создать комплексный тип для адреса, который должен содержать: страну, город, улицу, дом, квартиру.
- Ограничить список стран значениями: Russia, Ukraine, Moldova.
- Квартиры могут принимать значения от 1 до 200.
- Номера домов могут содержать символы.
- Проверить на адресе: Russia, Irkutsk, Engelsa, 10, 404

Задание 7

- Добавить к элементу адрес атрибут LANG, который должен принимать значения из списка RU, EN.

Пространства имен

Элементы и атрибуты, объявленные в элементе xsd:complexType, считаются локальными по отношению к составному типу. Локальные элементы могут использоваться только в пределах контекста, в котором они определены. Тут возникает интересный вопрос: а должно ли указываться пространство имен для локальных элементов/атрибутов в экземплярах документа. Поскольку локальные элементы и атрибуты всегда будут содержать родительский элемент (обычно глобальный элемент), определенный целевым пространством имен, то в этом нет необходимости.

Поэтому в XML Schema локальные элементы и атрибуты не должны указывать пространство имен по умолчанию.

```
<xsd:complexType name="AuthorType">
   <!-- compositor goes here -->
   <xsd:sequence>
      <xsd:element name="name" type="xsd:string"/>
      <xsd:element name="phone" type="tns:Phone"/>
   </xsd:sequence>
   <xsd:attribute name="id" type="tns:AuthorId"/>
</xsd:complexType>
<xsd:element name="author" type="tns:AuthorType"/>
```

```
<x:author xmlns:x="http://example.org/publishing"
   id="333-33-3333"
>
   <name>Aaron Skonnard</name>
   <phone>(801)390-4552</phone>
</x:author>
```


XML

Пространства имен

Однако XML Schema делает возможным явно контролировать, должен ли данный локальный элемент/атрибут содержать пространство имен или не должен, путем применения атрибута формы к xsd:element/xsd:attribute или атрибутов elementFormDefault/attributeFormDefault к xsd:schema.

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace="http://example.org/publishing"
   xmlns:tns="http://example.org/publishing"
   elementFormDefault="qualified"
   attributeFormDefault="qualified"
>
   ...
</xsd:schema>
```

```
<x:author xmlns:x="http://example.org/publishing"
   x:id="333-33-3333"
>
   <x:name>Aaron Skonnard</x:name>
   <x:phone>(801)390-4552</x:phone>
</x:author>
```

XSD индикаторы

Определение группы элементов (all, sequence, choice)
<xs:group name="groupname">
...
</xs:group>

Определение группы атрибутов
<xs:attributeGroup name="groupname">
...
</xs:attributeGroup>

Сложные типы

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>

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

Сложные типы

```
<xs:element name="employee" type="fullpersoninfo"/>
<xs:complexType name="personinfo">
  <xs:sequence>
    <xs:element name="firstname" type="xs:string"/>
    <xs:element name="lastname" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="fullpersoninfo">
  <xs:complexContent>
    <xs:extension base="personinfo">
      <xs:sequence>
        <xs:element name="address" type="xs:string"/>
        <xs:element name="city" type="xs:string"/>
        <xs:element name="country" type="xs:string"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Пустые элементы

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

<xs:element name="product">
  <xs:complexType>
    <xs:attribute name="prodid" type="xs:positiveInteger"/>
  </xs:complexType>
</xs:element>
```

От простого к сложному

Для расширения простого типа и создания нового типа можно использовать simpleContent. В качестве основы (base) указывается тот тип, который мы желаем изменить. С этим типом мы можем произвести расширение или сужение.

От простого к сложному

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:extension base="basetype">
        ....
        ....
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

От простого к сложному

```
<xs:element name="somename">
  <xs:complexType>
    <xs:simpleContent>
      <xs:restriction base="basetype">
        ....
        ....
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
</xs:element>
```

От простого к сложному

```
<xsd:element name="internationalPrice">
  <xsd:complexType>
   <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
     <xsd:attribute name="currency" type="xsd:string"/>
    </xsd:extension>
   </xsd:simpleContent>
  </xsd:complexType>
 </xsd:element>
```

Смешанное содержимое

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

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

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

Any

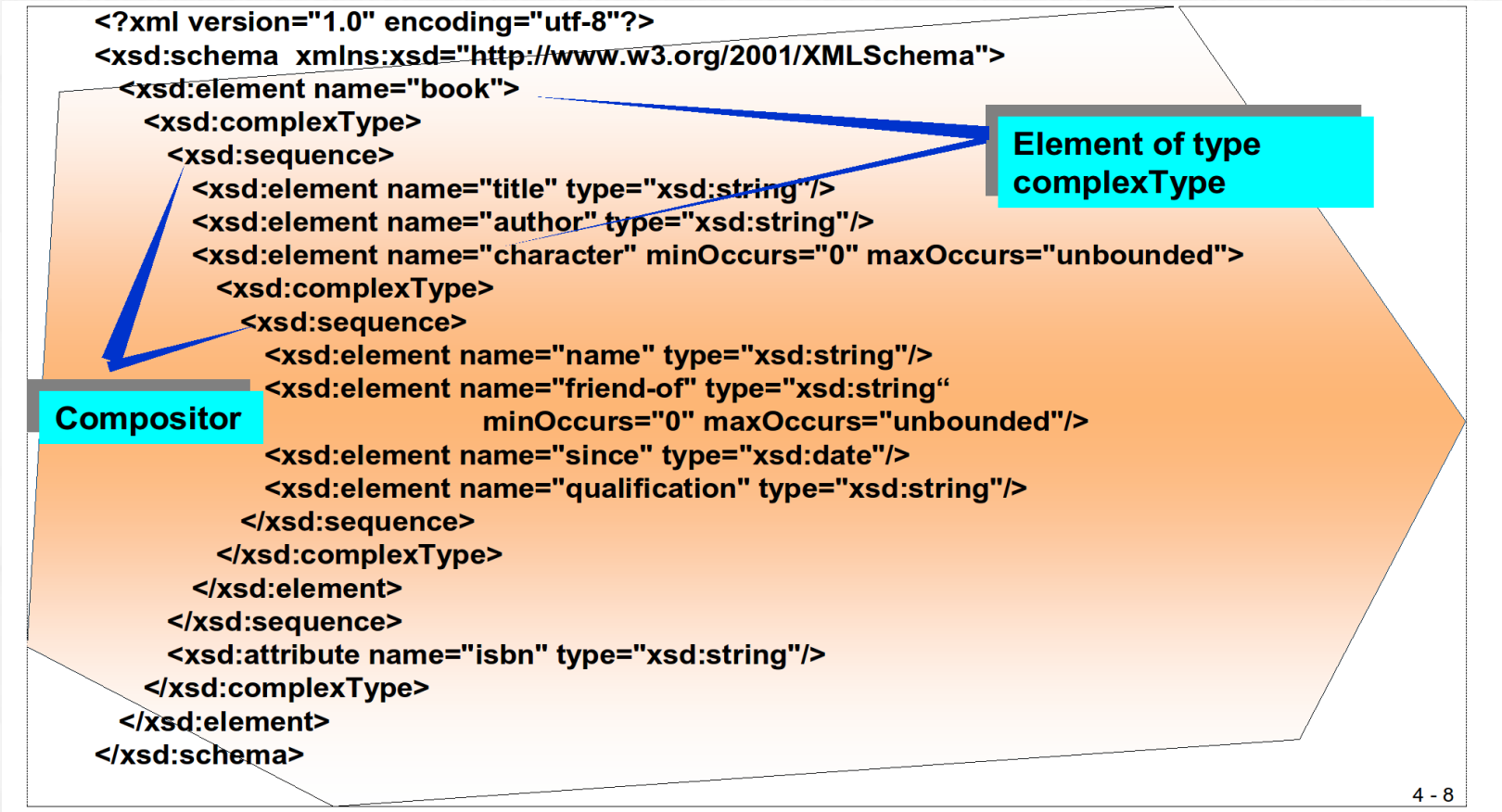
<xs:any minOccurs="0"/>

<xs:anyAttribute/>

Any

По умолчанию в составных типах есть модели закрытого содержимого (closed content models). Это означает, что в экземпляре могут появиться только указанные части. Однако с помощью групповых символов (wildcards) XML Schema дает возможность определять модель открытого содержимого (open content model). Использование xsd:any в составном типе означает, что в этом месте может появиться любой элемент, т.е. он играет роль структурного нуля для тех элементов, которые вы не можете прогнозировать. Также вы можете использовать xsd:anyAttribute для определения структурного нуля для атрибутов.

Пример


<?xml version="1.0" encoding="utf-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="book">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="author" type="xsd:string"/>
<xsd:element name="character" minOccurs="0" maxOccurs="unbounded">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="name" type="xsd:string"/>
<xsd:element name="friend-of" type="xsd:string“
minOccurs="0" maxOccurs="unbounded"/>
<xsd:element name="since" type="xsd:date"/>
<xsd:element name="qualification" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:sequence>
<xsd:attribute name="isbn" type="xsd:string"/>
</xsd:complexType>
</xsd:element>
</xsd:schema>
Element of type complexType
Compositor
4 - 8

Аннотации

Язык XML-схемы обеспечивает три элемента предназначенные для аннотации схемы. Содержимое этих элементов предназначено как для чтения человеком, так и для чтения приложением.

Элемент annotation может располагаться в начале других операторов языка XML-схемы, например, таких как schema, simpleType, и attribute.

```
<xsd:element name="internationalPrice">
 <xsd:annotation>
  <xsd:documentation xml:lang="en">
      element declared with anonymous type
  </xsd:documentation>
 </xsd:annotation>
 <xsd:complexType>
  <xsd:annotation>
   <xsd:documentation xml:lang="en">
       empty anonymous type with 2 attributes
   </xsd:documentation>
  </xsd:annotation>
  <xsd:complexContent>
   <xsd:restriction base="xsd:anyType">
    <xsd:attribute name="currency" type="xsd:string"/>
    <xsd:attribute name="value"    type="xsd:decimal"/>
   </xsd:restriction>
  </xsd:complexContent>
 </xsd:complexType>
</xsd:element>
```

Ссылки на описания

```
<xsd:element name="Item" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
   <xsd:sequence>
    <xsd:element   name="productName" type="xsd:string"/>
    <xsd:element   name="quantity">
     <xsd:simpleType>
      <xsd:restriction base="xsd:positiveInteger">
       <xsd:maxExclusive value="100"/>
      </xsd:restriction>
     </xsd:simpleType>
    </xsd:element>
    <xsd:element name="USPrice"  type="xsd:decimal"/>
    <xsd:element ref="comment"   minOccurs="0"/>
    <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
   </xsd:sequence>
   <xsd:attribute name="partNum"  type="SKU" use="required"/>
   <!-- add weightKg and shipBy attributes -->
   <xsd:attribute name="weightKg" type="xsd:decimal"/>
   <xsd:attribute name="shipBy">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:enumeration value="air"/>
      <xsd:enumeration value="land"/>
      <xsd:enumeration value="any"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:attribute>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="item" minOccurs="0" maxOccurs="unbounded">
 <xsd:complexType>
  <xsd:sequence>
   <xsd:element name="productName" type="xsd:string"/>
   <xsd:element name="quantity">
    <xsd:simpleType>
     <xsd:restriction base="xsd:positiveInteger">
      <xsd:maxExclusive value="100"/>
     </xsd:restriction>
    </xsd:simpleType>
   </xsd:element>
   <xsd:element name="USPrice"  type="xsd:decimal"/>
   <xsd:element ref="comment"   minOccurs="0"/>
   <xsd:element name="shipDate" type="xsd:date" minOccurs="0"/>
  </xsd:sequence>

  <!-- attributeGroup replaces individual declarations -->
  <xsd:attributeGroup ref="ItemDelivery"/>
 </xsd:complexType>
</xsd:element>
```

```
<xsd:attributeGroup name="ItemDelivery">
  <xsd:attribute name="partNum"  type="SKU" use="required"/>
  <xsd:attribute name="weightKg" type="xsd:decimal"/>
  <xsd:attribute name="shipBy">
    <xsd:simpleType>
     <xsd:restriction base="xsd:string">
      <xsd:enumeration value="air"/>
      <xsd:enumeration value="land"/>
      <xsd:enumeration value="any"/>
     </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:attributeGroup>
```

Отсутствие значения

Иногда желательно представить не отгруженное изделие, неизвестную или неподходящую информацию явно с помощью элемента, а не отсутствующим элементом. Например, это может быть полезным при работе с пустыми значениями ("Null") реляционной базы данных. Для этих целей в языке XML-схемы имеется Nil-механизм. Этот механизм позволяет элементу появляться с или без нулевого значения.

```
<xsd:element name="shipDate"
type="xsd:date" nillable="true"/>
```

```
<shipDate xsi:nil="true"></shipDate>
```

Многоязычность

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

Многоязычность

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

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="customer" type="custinfo"/>
<xs:element name="kunde" substitutionGroup="customer"/>
```

```
<customer>
  <name>John Smith</name>
</customer>
```

```
<kunde>
  <navn>John Smith</navn>
</kunde>
```

Многоязычность

```
<xs:element name="name" type="xs:string" block="substitution"/>
<xs:element name="navn" substitutionGroup="name"/>

<xs:complexType name="custinfo">
  <xs:sequence>
    <xs:element ref="name"/>
  </xs:sequence>
</xs:complexType>

<xs:element name="customer" type="custinfo"
block="substitution"/>
<xs:element name="kunde" substitutionGroup="customer"/>
```

Уникальность

В схеме так же возможно определить уникальность значений элемента или атрибута. Для этого используется элемент unique.

<unique
id=ID
name=NCName
any attributes
>

(annotation?,(selector,field+))

</unique>

Задание 8

- Создать в файл task5.xsd и описать в нем схему документа для task01.xml;
- Проверить документ на соответствие схеме.

Задание 9

- Создать в файл task6.xsd и описать в нем схему документа для task06.xml;
- Проверить документ на соответствие схеме.

Источники

1. Эдди С.Э. XML: справочник — СПб.: Питер, 2000
2. Материалы курса IT 581
3. Материалы курса по XML компьютерного центра обучения "Специалист" при МГТУ им. Н.Э. Баумана
4. Материалы W3Schools (http://www.w3schools.com/xml/)
5. Википедия (http://ru.wikipedia.org)
6. Наварро Э. XHTML: учебный курс — СПб.: Питер, 2001
7. Понимание XML Schema (http://www.vbnet.ru/articles/showarticle.aspx?id=151)
8. XML-СХЕМА (http://citforum.ru/internet/xml/scheme/)