Лекция 2.
XML - схема

DTD для XML Schemas

- schema

Определение типа

- complexType
- simpleType
- complexContent
- simpleContent
- extension
- restriction

Определение элементов

- element
- group
- all
- choice
- sequence

Открытый контент

- any
- anyAttribute

Определение атрибутов

- attribute
- attributeGroup

Ссылочные механизмы

- unique
- key
- keyref
- selector
- field

Комбинационные механизмы

- include
- import
- redefine
- notation

Передача инструкций или документация

- annotation
- appinfo
- documentation

Элемент Schemas

```
<!ELEMENT schema ((include | import | redefine | annotation)*,
  ((simpleType | complexType | element | attribute | attributeGroup | group | notation), (annotation)*)* )>

<!ATTLIST schema
  targetNamespace  %URIref;  #IMPLIED
  version   CDATA   #IMPLIED
  xmlns:xs %URIref; #FIXED http://www.w3.org/2001/XMLSchema'
  xmlns  CDATA  #IMPLIED
  finalDefault %complexDerivationSet; "
  blockDefault %blockSet;      "
  id   ID  #IMPLIED
  elementFormDefault %formValues; 'unqualified'
  attributeFormDefault %formValues; 'unqualified'
  xml:lang   CDATA   #IMPLIED
  %schemaAttrs;>
```

<!-- Декларация xmlns предназначена для схем автора -->

<!-- Атрибут id здесь и ниже предназначен для использования во внешних ссылках. Он не используется для ссылок от схемы на схему. -->

<!– тип является поименованным содержанием спецификации типа, которые используются при декларациях атрибута -->

Атрибуты элемента schema

id ID #IMPLIED

finalDefault %complexDerivationSet;
finalDefault: задание значения по умолчанию

blockDefault %blockSet;
blockDefault: задание значения по умолчанию

elementFormDefault %formValues; 'unqualified'
attributeFormDefault %formValues; 'unqualified‘

Допустимые значения для **elementFormDefault** (**attributeFormDefault**) - *qualified* и *unqualified*. Значение по умолчанию - *unqualified*.

Если **elementFormDefault (attributeFormDefault)**="*unqualified*”, то все локальные элементы (атрибуты) в документе не должны быть связаны с пространствами имен.

Если **elementFormDefault (attributeFormDefault)** ="*qualified*" то все локальные элементы (атрибуты) в документе должны быть связаны с пространствами имен.

Что использовать qualified или unqualified?

- В документе, независимо от того определяете Вы все элементы или только глобальные, все элементы разбираются.
- В чем различие при задании значения elementFormDefault “qualified” или “unqualified”?
- Случай 1: elementFormDefault="unqualified" и в документе определены только глобальные элементы
  - Pro: спрятана сложность пространства имен в схеме
  - Contra: если схема модифицируется, т.е. локальные декларации делаются глобальными, то все документы конфликтуют; пользователю надо помнить о том какие элементы глобальные и какие локальные.
- Случай 2: element elementFormDefault="qualified" и в документе определены все элементы
  - Pro: если в схеме локальные элементы преобразуются в глобальные, то нет конфликта документов; пользователю не надо помнить о том какие элементы глобальные, а какие локальные; для целей copyright желательно показывать пространства имен в документе.
  - Contra: показывается сложная структура пространств имен в документе

Элемент составной тип

<!ELEMENT **complexType** (annotation?, (simpleContent | complexContent | ((all | choice | sequence | group)?, ((attribute | attributeGroup)*,anyAttribute?))))>

<!ATTLIST complexType
name %NCName; #IMPLIED
id ID #IMPLIED
abstract %boolean; #IMPLIED
final %complexDerivationSet; #IMPLIED
block %complexDerivationSet; #IMPLIED
mixed (true | false) 'false'
%complexTypeAttrs;>

block: используется для запрещения типу элемента быть замененным производным типом

abstract: используется для создания порождающих типов, которые в документе не используются.

Атрибуты элемента complexType

Запрещение задания типов

- Иногда при создании типа необходимо запретить ему быть базовым типом для создания иных типов или только расширения или ограничения.
  - Причина: «Например, можно создать complexType и сделать его доступным для использования другим. Однако, может возникнуть потребность запрета на его изменение, скажем для случая копирайта."

<xsd:complexType name="Publication" final="#all" …> Тип не может быть редуцирован или расширен

<xsd:complexType name="Publication" final="restriction" …> Тип не может быть редуцирован

<xsd:complexType name="Publication" final="extension" …> Тип не может быть расширен

Атрибуты элемента complexType

mixed

- Содержанием элементов в документе могут быть элементы или данные
- Элемент, содержащий смесь элементов и (строковых) данных, называется смешанным контентом.
- Смешанный контент может иметь много приложений, Например, XSLT использует смешанный контент в правилах для шаблонов

```
<xsl:template match="Book">
    Название книги:
        <xsl:value-of select="Title/text()"/>
    Автор книги:
        <xsl:value-of select="Author/text()"/>
</xsl:template>
```

Заметим, что содержание xsl:элемента является смесью строковых данных и элементов.

Элементы complexContent, simpleContent, extension

```
<!ELEMENT complexContent (restriction|extension)>
<!ATTLIST complexContent
        mixed (true | false) #IMPLIED
        id    ID   #IMPLIED
        %complexContentAttrs;>

<!ELEMENT simpleContent (restriction|extension)>
<!ATTLIST simpleContent
        id    ID   #IMPLIED
        %simpleContentAttrs;>

<!ELEMENT extension ((all | choice | sequence | group),   ((attribute |
    attributeGroup), anyAttribute)))>
<!ATTLIST extension
        base  %QName; #REQUIRED
        id ID #IMPLIED
        %extensionAttrs;>
```

Элемент element

<!ELEMENT **element** ((annotation)?, (complexType | simpleType)?,
(unique | key | keyref)*)>

```
<!ATTLIST element
        name            %NCName;                #IMPLIED
        id              ID                      #IMPLIED
        ref             %QName;                 #IMPLIED
        type            %QName;                 #IMPLIED
        minOccurs       %nonNegativeInteger;    #IMPLIED
        maxOccurs       CDATA                   #IMPLIED
        nillable        %boolean;               #IMPLIED
        substitutionGroup  %QName;              #IMPLIED
        abstract        %boolean;               #IMPLIED
        final           %complexDerivationSet;  #IMPLIED
        block           %blockSet;              #IMPLIED
        default         CDATA                   #IMPLIED
        fixed           CDATA                   #IMPLIED
        form            %formValues;            #IMPLIED
        %elementAttrs;>
```

Атрибуты элемента Element

```
nillable        %boolean;                  #IMPLIED
abstract        %boolean;                  #IMPLIED
final           %complexDerivationSet;     #IMPLIED
block           %blockSet;                 #IMPLIED
form            %formValues;               #IMPLIED
```

- **form**: используется для отмены ***elementFormDefault*** (определенном в элементе ***schema***). Принимает два значения - *qualified* или *unqualified*. Если значение для ***form*** - qualified, то элемент должен быть отнесен к пространству имен в экземпляре документа.
- **abstract**: используется для указания на то, что элемент является собирательным именем для группы подстановки *substitutionGroup*. По умолчанию принимает значение *false*.
- **block**: используется для запрещения замены данного элемента на элемент из группы подстановки или запрещения типу элемента быть замененным производным типом.
- **nillable**: указывает на то, что значение элемента в документе может быть пусто, что указывается в значении атрибута xsi:null. Значение по умолчанию *false*.

Элемент группа элементов (group), неупорядоченный список (all)

```
<!ELEMENT group (annotation?, (all | choice | sequence)?)>
<!ATTLIST group
        name  %NCName; #IMPLIED
        ref %QName; #IMPLIED
        minOccurs   %nonNegativeInteger;   #IMPLIED
        maxOccurs   CDATA    #IMPLIED
        id   ID     #IMPLIED
        %groupAttrs;>

<!ELEMENT all ((annotation)?, (element)*)>
<!ATTLIST all
        minOccurs   (1) #IMPLIED
        maxOccurs   (1) #IMPLIED
        id  ID  #IMPLIED
        %allAttrs;>
```

Элементы (выбор и упорядоченный список (choice, sequence)

```
<!ELEMENT choice ((annotation)?, (element| group| (choice | sequence) |
    any)*)>
<!ATTLIST choice
        minOccurs   %nonNegativeInteger;   #IMPLIED
        maxOccurs   CDATA  #IMPLIED
        id  ID  #IMPLIED
        %choiceAttrs;>

<!ELEMENT sequence ((annotation)?, (element| group| (choice | sequence) |
    any)*)>
<!ATTLIST sequence
        minOccurs   %nonNegativeInteger;   #IMPLIED
        maxOccurs   CDATA   #IMPLIED
        id ID   #IMPLIED
        %sequenceAttrs;>
```

Элементы произвольный элемент и произвольный атрибут (any, anyAttribute)

```
<!ELEMENT any (annotation)?>
<!ATTLIST any
        namespace  CDATA                '##any'
        processContents (skip|lax|strict)     'strict'
        minOccurs       %nonNegativeInteger;   '1'
        maxOccurs       CDATA                  '1'
        id  ID   #IMPLIED
        %anyAttrs;>
<!– пространство имен специфицируется следующим образом:
      ##any      - - любой не конфликтующий корректный XML контент
      ##other    - - любой не конфликтующий корректный XML контент из пространства имен иного чем
    targetNamespace
      ##local    - - любой не проверяемый не конфликтующий корректный XML контент/атрибут-->
<!ELEMENT anyAttribute (annotation)?>
<!ATTLIST anyAttribute
        namespace       CDATA             '##any'
        processContents (skip|lax|strict)  'strict'
        id              ID                #IMPLIED
        %anyAttributeAttrs;>
   пространство имен специфицируется  также как 'any'
   simpleType only if no type|ref attribute
   ref not allowed at top level, name iff at top level
```

wfxml спецификация контента (well-formed XML)

Элемент атрибут (attribute)

```
<!ELEMENT attribute ((annotation)?, (simpleType)?)>
<!ATTLIST attribute
  name      %NCName;      #IMPLIED
  id        ID            #IMPLIED
  ref       %QName;       #IMPLIED
  type      %QName;       #IMPLIED
  use  (prohibited|optional|required)  #IMPLIED
  default   CDATA         #IMPLIED
  fixed     CDATA         #IMPLIED
  form      %formValues;  #IMPLIED
       %attributeAttrs;>
```

type и *ref* взаимно исключаемы. *name* и *ref* взаимно исключаемы, допустимо либо одно, либо другое. Значение по умолчанию для *use* - optional. *default* и *fixed* взаимно исключаемы. *type* и *simpleType* взаимно исключаемы

Элемент группа атрибутов (attributeGroup)

<!ELEMENT **attributeGroup** (annotation?, (attribute | attributeGroup)*,
anyAttribute?)>

<!ATTLIST %attributeGroup;

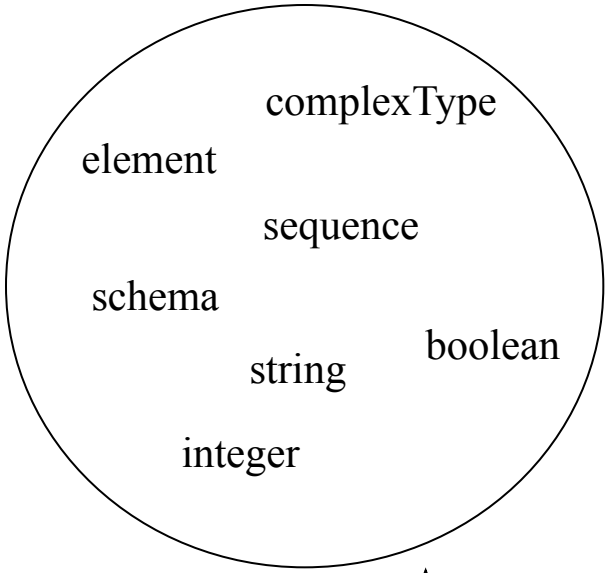
name %NCName; #IMPLIED

id ID #IMPLIED

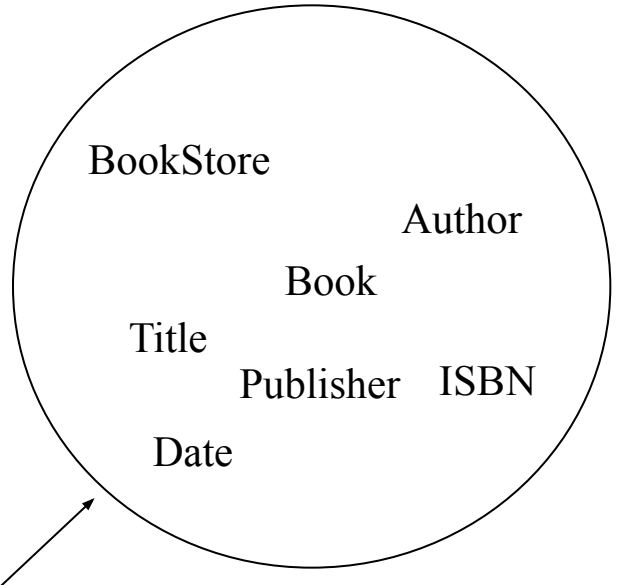
ref %QName; #IMPLIED

%attributeGroupAttrs;>

http://www.w3.org/2001/XMLSchema



http://www.books.org (*targetNamespace*)



Это словарь с помощью которого XML схема обеспечивают создание нового словаря

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore-Книжный-магазин">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book-Книга">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title-Название" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author-Автор" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date-Дата-публикации" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher-Издатель" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title-Название " type="xsd:string"/>
  <xsd:element name="Author-Автор " type="xsd:string"/>
  <xsd:element name="Date-Дата-публикации " type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher-Издатель " type="xsd:string"/>
</xsd:schema>
```

BookStore.xsd

xsd = Xml-Schema Definition

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

<!ELEMENT BookStore (Book+)>

<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>

<!ELEMENT Title (#PCDATA)>

<!ELEMENT Author (#PCDATA)>

<!ELEMENT Date (#PCDATA)>

<!ELEMENT ISBN (#PCDATA)>

<!ELEMENT Publisher (#PCDATA)>

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Все XML схемы имеют тег "schema" В качестве корневого элемента.

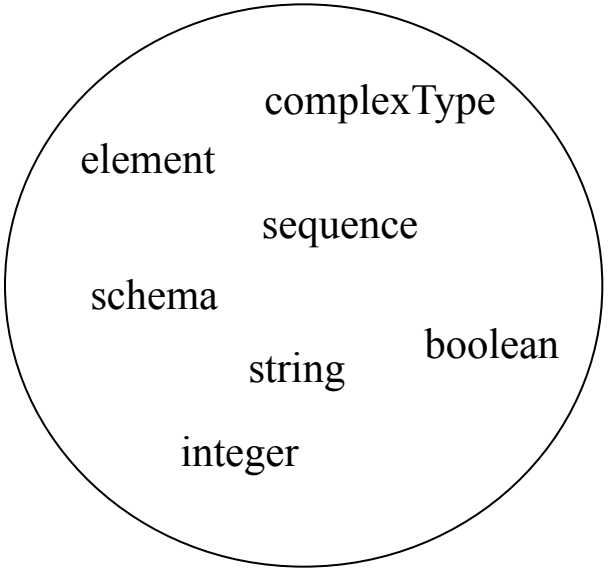
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Элементы и типы данных, используемые для построения схем
- schema
- element
- complexType
- sequence
- string

принадлежат пространству имен http://…/XMLSchema

Пространство имен XMLSchema

http://www.w3.org/2001/XMLSchema



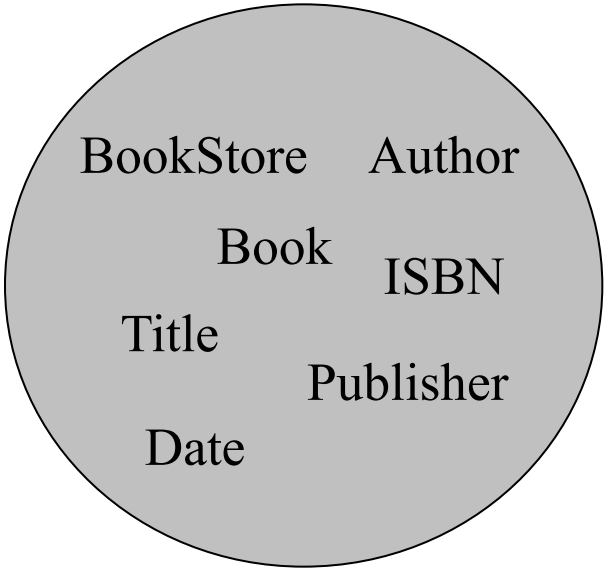
```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns="http://www.books.org"
            elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Указывает на то, что элементы, определяемые схемой
- BookStore
- Book
- Title
- Author
- Date
- ISBN
- Publisher

принадлежат пространству имен http://www.books.org

Пространство имен Books (targetNamespace)

http://www.books.org (targetNamespace)



```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

Пространство имен по умолчанию http://www.books.org совпадает с targetNamespace!

Вот ссылка на декларацию элемента Book.
В каком пространстве имен Book? Так как приставка отсутствует, то элемент Book в пространстве имен по умолчанию, т.е. targetNamespace! Т.о., это ссылка на декларацию элемента Book в данной схеме.

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Book" minOccurs="1"
maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Book">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="Title" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Author" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Date" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="ISBN" minOccurs="1" maxOccurs="1"/>
        <xsd:element ref="Publisher" minOccurs="1" maxOccurs="1"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="Title" type="xsd:string"/>
  <xsd:element name="Author" type="xsd:string"/>
  <xsd:element name="Date" type="xsd:string"/>
  <xsd:element name="ISBN" type="xsd:string"/>
  <xsd:element name="Publisher" type="xsd:string"/>
</xsd:schema>
```

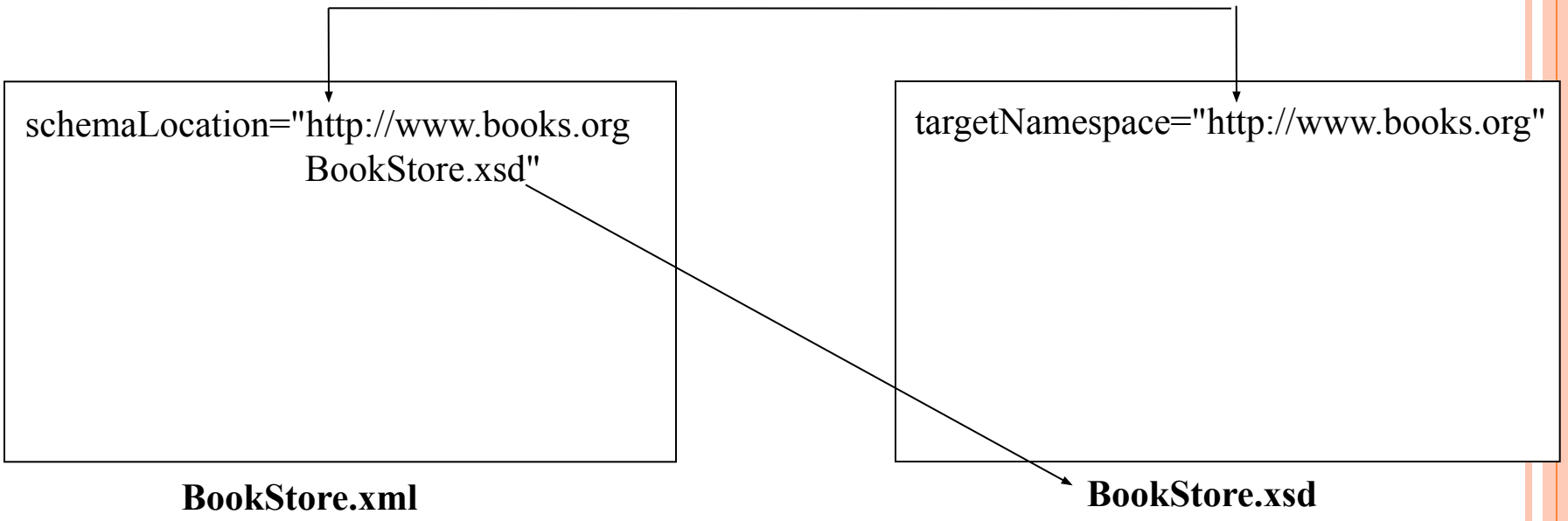
Это директива любому элементу соответствующему данной схеме:
Любые элементы, используемые в документе, которые декларированы в этой схеме, должны быть специфицированы по пространствам имен.

Ссылка на схему в XML документе

```
<?xml version="1.0"?>
<BookStore xmlns ="http://www.books.org" (1)                    (3)
           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
           xsi:schemaLocation="http://www.books.org
                               BookStore.xsd"; (2)
    <Book>
        <Title>My Life and Times</Title>
        <Author>Paul McCartney</Author>
        <Date>July, 1998</Date>
        <ISBN>94303-12021-43892</ISBN>
        <Publisher>McMillin Publishing</Publisher>
    </Book>
    ...
</BookStore>
```

1. Использование пространства имен по умолчанию указывает парсеру на то, что все элементы документа относятся к пространству имен *http://www.books.org*.
2. schemaLocation указывает парсеру на то, что в пространстве имен *http://www.books.org* определена схема BookStore.xsd (i.e., **schemaLocation** содержит пару значений).
3. указывает парсеру на то, что атрибут schemaLocation определен в пространстве имен XMLSchema-instance.

Ссылка на схему в XML документе

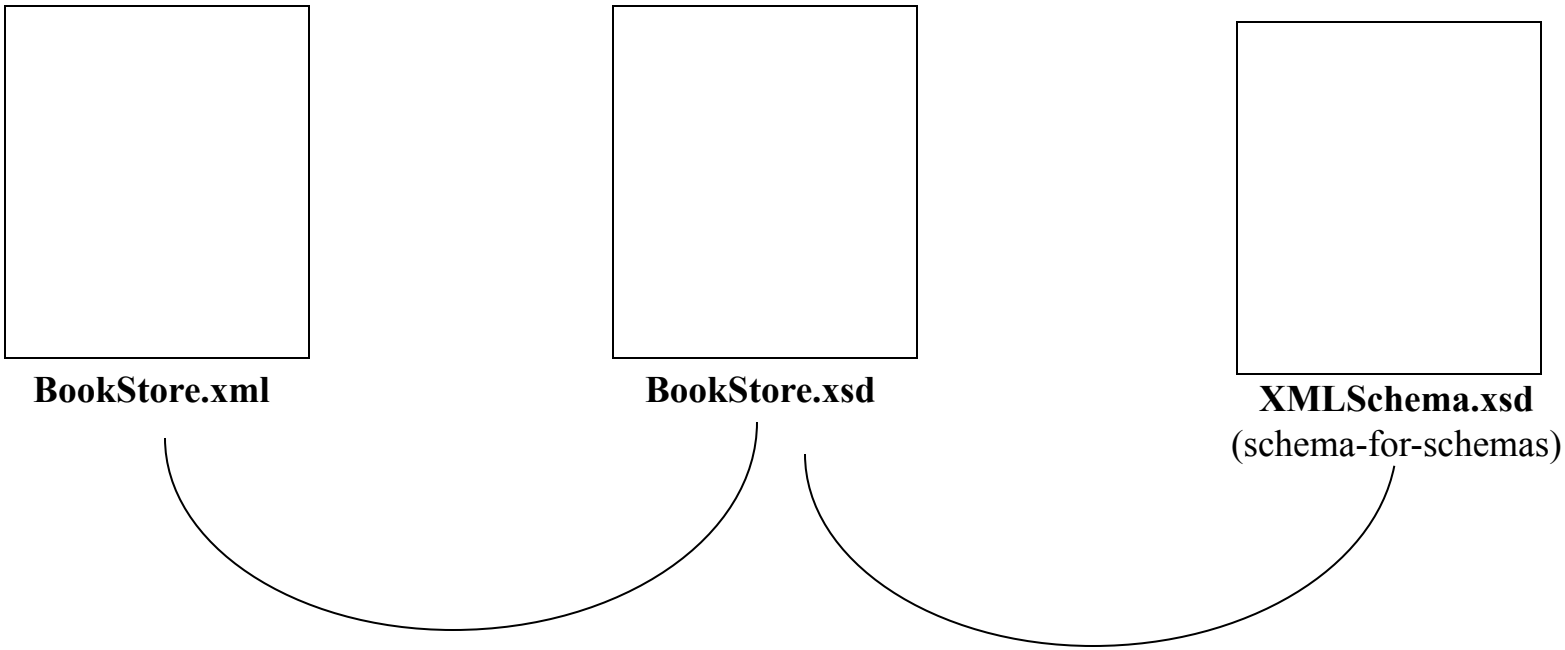


- *использует* элементы из пространства имен http://www.books.org

- *определяет* элементы в пространстве имен http://www.books.org

Схема *определяет* новый словарь. Документы *используют* этот словарь.

Многоуровневость проверки

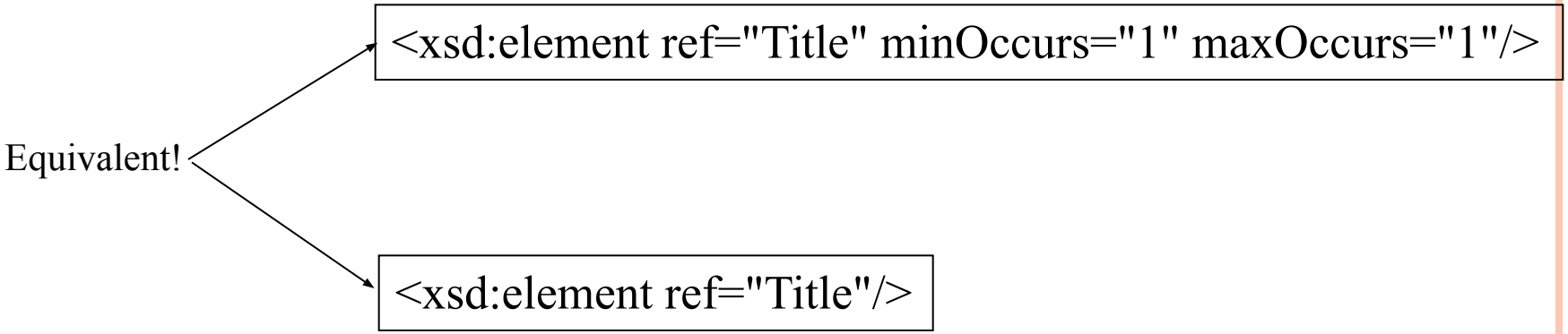


Проверка на то, что xml документ Соответствует правилам, описанным в BookStore.xsd

Проверяем, что BookStore.xsd является корректной схемой, i.e., он соответствует правилам описанным в schema-for-schemas

Значения по умолчанию для minOccurs и maxOccurs

- Значение по умолчанию для minOccurs - "1"
- Значение по умолчанию для maxOccurs - "1"



```
<?xml version="1.0"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.books.org"
            xmlns:bk="http://www.books.org"
            elementFormDefault="qualified">
   <element name="BookStore">
      <complexType>
         <sequence>
            <element ref="bk:Book" maxOccurs="unbounded"/>
         </sequence>
      </complexType>
   </element>
   <element name="Book">
      <complexType>
         <sequence>
            <element ref="bk:Title"/>
            <element ref="bk:Author"/>
            <element ref="bk:Date"/>
            <element ref="bk:ISBN"/>
            <element ref="bk:Publisher"/>
         </sequence>
      </complexType>
   </element>
   <element name="Title" type="string"/>
   <element name="Author" type="string"/>
   <element name="Date" type="string"/>
   <element name="ISBN" type="string"/>
   <element name="Publisher" type="string"/>
</schema>
```

Заметим, что http://…/XMLSchema является пространством имен по умолчанию. Следовательно, нет необходимости использовать приставки для
- schema
- element
- complexType
- sequence
- string

Линейные декларации элемента

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book"  maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:string"/>
              <xsd:element name="ISBN" type="xsd:string"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Все декларации элементов помещены внутрь и исчезла возможность ссылаться на эти элементы. Схема стала более компактной!

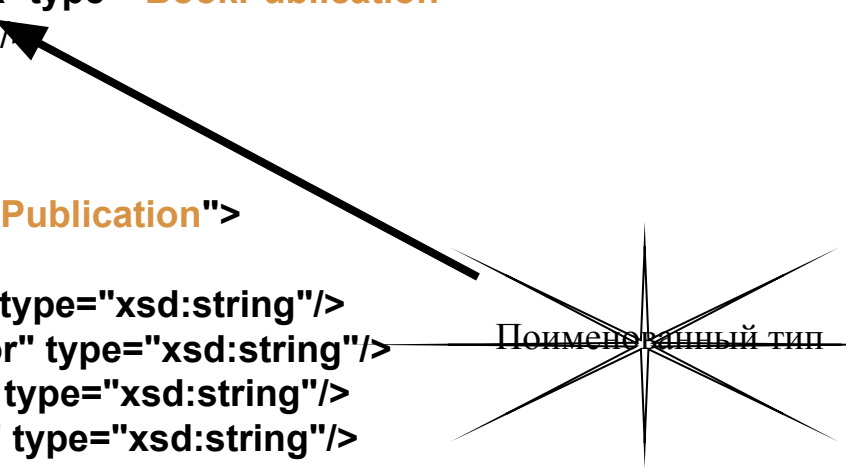
Такой способ создания схемы – вкладывание во внутрь – называют матрешкой

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
   <xsd:element name="BookStore">
      <xsd:complexType>
         <xsd:sequence>
            <xsd:element name="Book"  maxOccurs="unbounded">
               <xsd:complexType>
                  <xsd:sequence>
                     <xsd:element name="Title" type="xsd:string"/>
                     <xsd:element name="Author" type="xsd:string"/>
                     <xsd:element name="Date" type="xsd:string"/>
                     <xsd:element name="ISBN" type="xsd:string"/>
                     <xsd:element name="Publisher" type="xsd:string"/>
                  </xsd:sequence>
               </xsd:complexType>
            </xsd:element>
         </xsd:sequence>
      </xsd:complexType>
   </xsd:element>
</xsd:schema>
```

Анонимные типы

Поименованные типы

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book" type="BookPublication"
          maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:complexType name="BookPublication">
    <xsd:sequence>
      <xsd:element name="Title" type="xsd:string"/>
      <xsd:element name="Author" type="xsd:string"/>
      <xsd:element name="Date" type="xsd:string"/>
      <xsd:element name="ISBN" type="xsd:string"/>
      <xsd:element name="Publisher" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```



Преимущество расщепления декларации элемента Book и введения поименованного типа состоит в том, что он может использоваться другими элементами.

Заметим, что:

```
<xsd:element name="A" type="foo"/>
<xsd:complexType name="foo">
  <xsd:sequence>
      <xsd:element name="B" …/>
      <xsd:element name="C" …/>
  </xsd:sequence>
</xsd:complexType>
```

Элемент A ссылается на complexType foo.

эквивалентно:

```
<xsd:element name="A">
  <xsd:complexType>
      <xsd:sequence>
          <xsd:element name="B" …/>
          <xsd:element name="C" …/>
      </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

Элемент A содержит декларацию complexType внутри своей декларации.

Атрибут **type** или дочерний элемент **complexType**

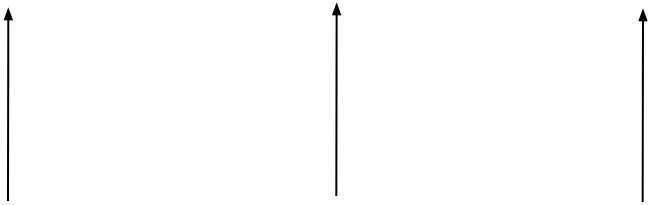
- Декларация элемента может иметь или атрибут *type*, или дочерний элемент *complexType*.

```
<xsd:element name="A" type="foo">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

Декларация элементов. Резюме.

1

<xsd:element name="*name*" type="*type*" minOccurs="*int*" maxOccurs="*int*"/>



Простой тип
(e.g., xsd:string)
или
a complexType
(e.g., BookPublication)

Неотриц. целое

Неотрицательное целое
или "unbounded"

Note: minOccurs и maxOccurs можно использовать в локальных декларациях элементов

2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

Типы данных

- Встроенные типы данных (i.e., известные типы для парсеров схемы)
- Этот тип данных используется для представления дня (год-месяц-день)
- Элементы с типом date должны иметь форму: CCYY-MM-DD
  - диапазон CC : 00-99
  - диапазон YY : 00-99
  - диапазон MM: 01-12
  - диапазон DD:
    - 01-28 если месяц 2
    - 01-29 если месяц 2 и gYear высокосный
    - 01-30 если месяц 4, 6, 9 или 11
    - 01-31 если месяц 1, 3, 5, 7, 8, 10 или 12
  - Example: 1999-05-31 представляет May 31, 1999

Тип данных gYear

- Встроенный тип (Григорианский календарь)
- Элементы с типом gYear должны иметь форму : CCYY
  - диапазон CC : 00-99
  - диапазон YY : 00-99
  - Пример: 1999 указывает год gYear 1999

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.books.org"
                xmlns="http://www.books.org"
                elementFormDefault="qualified">
  <xsd:simpleType name="ISBNType">
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:element name="BookStore">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Book"  maxOccurs="unbounded">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="Title" type="xsd:string"/>
              <xsd:element name="Author" type="xsd:string"/>
              <xsd:element name="Date" type="xsd:gYear"/>
              <xsd:element name="ISBN" type="ISBNType"/>
              <xsd:element name="Publisher" type="xsd:string"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Определение нового типа данных ISBNType.

Декларация Date с типом gYear, и ISBN с типом ISBNType (defined above)

```
<xsd:simpleType name="ISBNType">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
   </xsd:restriction>
</xsd:simpleType>
```

Продекларирован новый тип ISBNType. Он является ограничительной формой типа string. Элементы такого типа должны соответствовать следующим образцам:

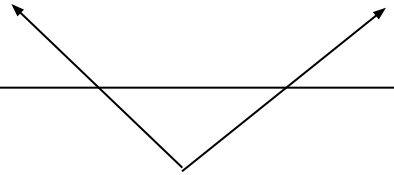
- Первый шаблон: 1 цифра - 5 цифр - 3 цифры - 1 цифра
- Второй шаблон: 1 - 3 - 5 - 1
- Третий шаблон: 1 - 2 - 6 - 1

Эти образцы специфицируются с помощью *Regular Expressions*.

Эквивалентные выражения

```
<xsd:simpleType name="ISBNType">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
      <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
   </xsd:restriction>
</xsd:simpleType>
```

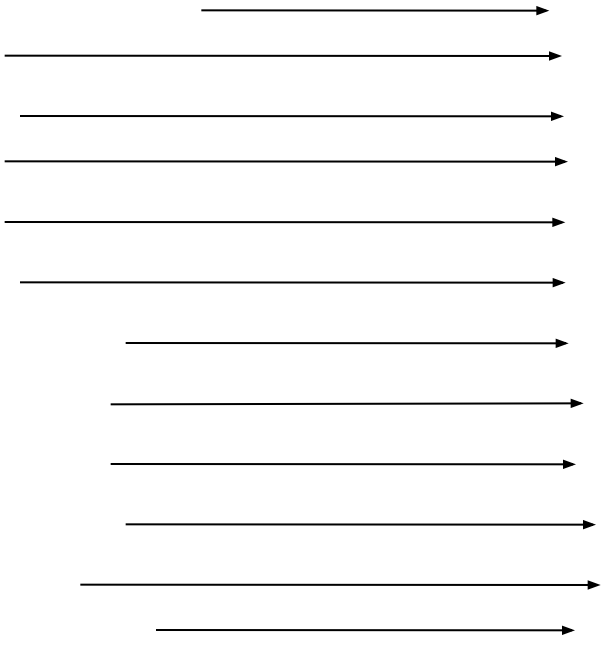
```
<xsd:simpleType name="ISBNType">
   <xsd:restriction base="xsd:string">
      <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1} | \d{1}-\d{3}-\d{5}-\d{1} | \d{1}-\d{2}-\d{6}-\d{1}"/>
   </xsd:restriction>
</xsd:simpleType>
```



Вертикальная линия означает “или"

Встроенные типы данных

- Примитивы типов данных
  - String
  - boolean
  - decimal
  - float
  - double
  - duration
  - dateTime
  - time
  - date
  - gYearMonth
  - gYear
  - gMonthDay



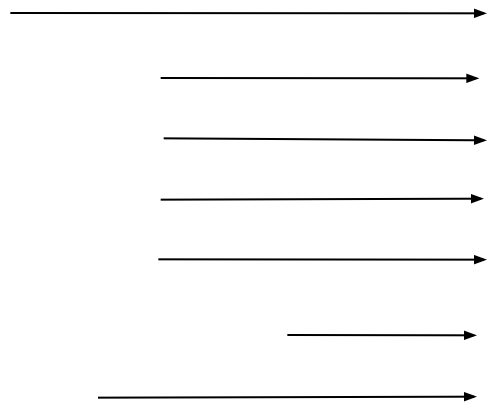
Атомные, встроенные

- **"Hello World"**
- **{true, false, 1, 0}**
- **7.08**
- **12.56E3, 12, 12560, 0, -0, INF, -INF, NAN**
- **12.56E3, 12, 12560, 0, -0, INF, -INF, NAN**
- **P1Y2M3DT10H30M12.3S**
- format: *CCYY-MM-DDThh:mm:ss*
- format: *hh:mm:ss.sss*
- format: *CCYY-MM-DD*
- format: *CCYY-MM*
- format: *CCYY*
- format: *--MM-DD*

Замечание: 'T' – разделитель дня от часов
INF = infinity
NAN = not-a-number

Встроенные типы данных

- Примитивы типов данных
  - gDay
  - gMonth
  - hexBinary
  - base64Binary
  - anyURI
  - Qname
  - NOTATION



Атомные, встроенные

- format: ---*DD* (note the 3 dashes)
- format: --*MM*--
- a hex string
- a base64 string
- **http://www.xfront.com**
- a namespace qualified name
- a NOTATION from the XML spec

Встроенные типы данных

Производные типы данных	Подтипы примитивов типов данных
normalizedString	A string without tabs, line feeds, or carriage returns
Token	String w/o tabs, l/f, leading/trailing spaces, consecutive spaces
language	any valid xml:lang value, e.g., **EN**, **FR**, …
IDREFS	Должно использоваться только с атрибутами
ENTITIES	Должно использоваться только с атрибутами
NMTOKEN	Должно использоваться только с атрибутами
NMTOKENS	Должно использоваться только с атрибутами
Name	**part** (no namespace qualifier)
NCName	Должно использоваться только с атрибутами
ID, IDREF	Должно использоваться только с атрибутами
ENTITY	Должно использоваться только с атрибутами
integer	**456**
nonPositiveInteger	negative infinity to 0

Встроенные типы данных

- Производные типы
  - negativeInteger → negative infinity to -1
  - Long → **-9223372036854775808** to **9223372036854775807**
  - int → **-2147483648** to **2147483647**
  - short → **-32768** to **32767**
  - byte → **-127** to **128**
  - nonNegativeInteger → 0 to infinity
  - unsignedLong → **0** to **18446744073709551615**
  - unsignedInt → **0** to **4294967295**
  - unsignedShort → **0** to **65535**
  - unsignedByte → **0** to **255**
  - positiveInteger → **1** to infinity

- Подтипы примитивов типов данных

Note: следующие типы можно использовать с атрибутами:
ID, IDREF, IDREFS, NMTOKEN, NMTOKENS, ENTITY, and ENTITIES.

<xsd:complexType> или <xsd:simpleType>?

- Когда использовать элемент complexType или simpleType?
  - Используйте элемент complexType тогда, когда хотите определить дочерние элементы и/или атрибуты элемента
  - Используйте элемент simpleType тогда, когда хотите создать новый тип который модифицирует встроенный тип данных (string, date, gYear, etc)

Элемент с простым типом

Пример. Создать декларацию элемента elevation.
Декларировать элемент elevation как целое в интервале -1290 до 29035

```
<elevation>5240</elevation>
```

Один из способов декларации:

```
<xsd:simpleType name="EarthSurfaceElevation">
  <xsd:restriction base="xsd:integer">
    <xsd:minInclusive value="-1290"/>
    <xsd:maxInclusive value="29035"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:element name="elevation" type="EarthSurfaceElevation"/>
```

Элемент с простым типом

Другой способ:

```
<xsd:element name="elevation">
   <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
         <xsd:minInclusive value="-1290"/>
         <xsd:maxInclusive value="29035"/>
      </xsd:restriction>
   </xsd:simpleType>
</xsd:element>
```

Определение (анонимное) simpleType внутри элемента.

Недостаток такого подхода состоит в том, что данный simpleType нельзя использовать в других элементах.

Декларация элементов. Резюме.

1

```
<xsd:element name="name" type="type" minOccurs="int" maxOccurs="int"/>
```

2

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:complexType>
        …
    </xsd:complexType>
</xsd:element>
```

3

```
<xsd:element name="name" minOccurs="int" maxOccurs="int">
    <xsd:simpleType>
        <xsd:restriction base="type">
        …
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
```

Литература

1. *XML-схема. Часть 0: пример.*
2. *Расширяемый язык разметки (XML) 1.0 (вторая редакция).*
3. *Пространства имен в XML*
4. *Язык XML - практическое введение, Александр Печерский*
5. *Язык XML - практическое введение. Часть 2, Александр Печерский*
6. **Extensible Markup Language (XML) 1.1 (Second Edition)**
7. **Namespaces in XML 1.1 (Second Edition)**
8. **XML Schema Part 0: Primer Second Edition**
9. **XML Schema Part 1: Structures Second Edition**
10. **XML Schema Part 2: Datatypes Second Edition**