

Пример DTD-определения

Пример иллюстрирует DTD-определение для документа "Журнал". Корневой элемент `Journal`, включает в себя подэлементы `Title` и `Article`, причем элемент `Title` является обязательным, может встречаться в документе только один раз и содержит любую символьную информацию. Элемент `Article` также обязательный, но может встречаться несколько раз, содержит следующие подэлементы: `Author` - обязательный элемент, может встречаться несколько раз; `Title` - обязательный, встречается один раз; `Annotation` - необязательный элемент, может встречаться только один раз; `KeyWord` - является необязательным, может встречаться несколько раз. Кроме того, элемент `Article` содержит атрибуты, которые объявляются сразу после объявления самого элемента: `BookID` - обязательный атрибут, содержит любую символьную информацию; `References` - необязательный атрибут, содержит любую символьную информацию; `Index` - атрибут может принимать одно из двух указанных значений, значение по умолчанию "no"; `Field` - фиксированный атрибут, может содержать содержит любую символьную информацию, при этом для него установлено значение "technical".

Элемент `Author` является пустым, но содержит следующие атрибуты, которые содержат символьную информацию: `FirstName` - обязательный, `MiddleName` - необязательный; `LastName` - обязательный. Элементы `Annotation` и `KeyWord` не имеют ни вложенных элементов, ни атрибутов, могут содержать любую символьную информацию.

```
<!ELEMENT Journal (Title, Article+)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Article (Author+, Title, Annotation?, KeyWord*)>
<!ATTLIST Article
    BookID CDATA #REQUIRED
    References CDATA #IMPLIED
    Index (yes | no) "no"
    Field CDATA #FIXED "technical"
>
<!ELEMENT Author EMPTY>
<!ATTLIST Author
    FirstName CDATA #REQUIRED
    MiddleName CDATA #IMPLIED
    LastName CDATA #REQUIRED
>
```

```
<!ELEMENT Annotation (#PCDATA)>
<!ELEMENT KeyWord (#PCDATA)>
```

Пример XML-документа на основе приведенного DTD-определения

Элемент Journal [Журнал] включает с себя два подэлемента - Title [Заголовок] со значением «Телематика» и Article [Статья]. Элемент Article состоит из набора подэлементов: Author - автор; Title - заголовок статьи; Annotation - аннотация; KeyWord - ключевые слова, а также атрибутов: BookID - идентификационный номер; References - ссылки. Элемент Author в свою очередь имеет атрибуты: FirstName - имя; MiddleName - отчество; LastName – фамилия, других данных не содержит.

```
<?xml version="1.0" encoding="windows-1251"?>
<!DOCTYPE Journal SYSTEM "Journal.dtd">
<Journal>
  <Title>Телематика</Title>
  <Article BookID="1" References="6">
    <Author FirstName="Андрей" MiddleName="Петрович" LastName="Иванов"/>
    <Title>XML-реляционные преобразования</Title>
    <Annotation>В статье рассматривается задача построения реляционной
модели на основе XML-документов.</Annotation>
    <KeyWord>СУБД</KeyWord>
    <KeyWord>XML</KeyWord>
  </Article>
</Journal>
```

Пример описания структуры XML-документа на основе XML-схемы

В строке 2 значение атрибута xmlns с префиксом xs является ссылкой на стандартное пространство имен <http://www.w3.org/2001/XMLSchema> для экземпляра XML схемы. Документ "Журнал" состоит из корневого элемента Journal и подэлементов Title и Article. Подэлемент Article содержит в свою очередь другие подэлементы, причем подэлементы Title, Annotation, KeyWord не содержат подэлементов, но имеют атрибуты. Элементы, которые содержат подэлементы или имеют атрибуты, называют *элементами комплексного типа* (complexType), тогда как элементы, которые содержат числа (строки, даты, и т.д.), но не содержат подэлементов или атрибутов, называются *элементами простого типа* (simpleType). Атрибуты всегда представляют собой элементы простого типа. Элемент

Article имеет атрибуты BookID, References, Index, а его подэлемент Author обладает атрибутами FirstName, MiddleName, LastName.

```
<?xml version="1.0" encoding="Windows-1251"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified">
  <xs:element name="Journal">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Title"/>
        <xs:element ref="Article" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Title" type="xs:string"/>
  <xs:element name="Article">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Author" maxOccurs="unbounded"/>
        <xs:element ref="Title"/>
        <xs:element ref="Annotation"/>
        <xs:element ref="KeyWord" minOccurs="0"
maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="BookID" type="xs:unsignedInt"/>
      <xs:attribute name="References" type="xs:unsignedInt"/>
      <xs:attribute name="Index" default="no">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="yes"/>
            <xs:enumeration value="no"/>
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="Field" fixed="technical"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="Author">
```

```

<xs:complexType>
  <xs:attribute name="FirstName" type="xs:string" use="required"/>
  <xs:attribute name="MiddleName" type="xs:string"/>
  <xs:attribute name="LastName" type="xs:string" use="required"/>
</xs:complexType>
</xs:element>
<xs:element name="Annotation" type="xs:string"/>
<xs:element name="KeyWord" type="xs:string"/>
</xs:schema>

```

Ниже в таблице представлено описание основных параметров атрибутов, используемых в XMLсхема.

| Параметры атрибутов | Описание |
|---------------------|--|
| abstract | Указывает, может ли составной тип использоваться в экземпляре документа. Если значение равно true, элемент не может непосредственно использовать данный составной тип, но может использовать составной тип, производный от данного. Значение по умолчанию - false. Необязательный. |
| block | Способ получения производных типов. Атрибут block запрещает использование составного типа, полученного от данного указанным способом, вместо данного составного типа. Может принимать значение #all или список, являющийся подмножеством расширения или ограничения. Атрибут block используется только если при проверке экземпляр документа переопределяет исходный тип элемента с использованием атрибута schema-instance:type. Атрибут block может запретить элементам заменять свой исходный тип на составной тип, полученный из исходного расширением и/или ограничением. |
| base | Имя встроенного типа данных, элемента simpleType или complexType с простым наполнением. Значение base обязательно и должно быть полным именем (QName). |
| id | Идентификатор этого элемента. Значение id должно иметь тип ID и быть уникальным в документе, содержащем этот элемент. Необязательно. |
| name | Имя группы атрибутов, включенных в элемент complexType. Указывается локальное имя группы так, как оно указано в спецификации пространств |

| | |
|-----------|--|
| | имен XML. Атрибуты name и ref не могут быть представлены вместе. Атрибут name может быть представлен, только если группа атрибутов является дочерней по отношению к элементу schema. Необязательно. |
| ref | Ссылочное имя группы атрибутов, включенных в элемент complexType. Это значение должно быть полным именем (QName). Необязательно. |
| maxOccurs | Наибольшее число раз, которое данный элемент может встретиться в файле. Значение должно быть целым числом больше единицы. Необязательно. |
| minOccurs | Наименьшее число раз, которое данный элемент может встретиться в файле. Чтобы указать, что данный элемент не обязателен, устанавливается значение «0», по умолчанию значение «1». Необязательно. |
| default | Атрибут со значением, определенным по умолчанию, может появляться или не появляться в документе. |
| final | Атрибут со значением restriction предотвращает образования новых типов методом ограничения. Значение #all предотвращает образование новых типов вообще. Значение extension предотвращает образования новых типов методом расширения. |
| fixed | Используется, чтобы указать, что атрибут или элемент могут принимать фиксированные значения. |
| type | Тип идентифицируется с помощью атрибута type, который принадлежит именному пространству языка XML-схемы. |
| value | Позволяет устанавливать значение какого-либо типа. |
| use | Значение параметра атрибута может быть required - обязателен, optional - необязателен, или prohibited - запрещен. |

Используемые типы данных:

| Типы данных | Описание |
|-------------|---|
| string | Позволяет специфицировать символьные данные в виде строк. Строка определяется как конечная последовательность символов. |
| boolean | Предназначен для реализации логических значений. Является перечисляемым типом с допустимыми значениями true и false. |
| integer | Предназначен для хранения целочисленных значений. |
| binary | Предоставляет возможность хранить и обрабатывать данные в двоичном |

| | |
|--------------|---|
| | коде. Применяется для реализации любых объектных вставок [графика, звук, видеоклипы]. |
| uriReference | Позволяет задавать URI в качестве содержимого элемента. |
| unsignedInt | Производный от типа unsignedLong, данные которого лежат в промежутке от 0 до 4294967295. Тип unsignedLong - вдвое экономичнее. |
| ID | Содержит значения уникальных идентификаторов экземпляра объекта в XML-документе. |
| Name | Представляет имена в XML. Имя, начинающееся с буквы, подчеркивания или двоеточия и продолжающееся знаками имени [буквами, цифрами и другими знаками]. Этот тип данных является производным от типа token. |
| QName | Представляет названия в пространстве имен XML в виде полных имен [qualified names], содержащих единственный символ двоеточия, делящий такое имя на префикс пространства имен и локальную часть. |
| NCName | Представляет имена, не начинающиеся с двоеточия. Идентичен типу Name, за тем исключением, что он не может начинаться с двоеточия. |
| TOKEN | Представляет размеченные строки. Этот тип данных является производным от типа string. |
| EMTOKEN | Набор знаков имени [букв, цифр и других знаков] в любой комбинации. В отличие от типов Name и NCName, он не имеет ограничений на начальный знак. Этот тип данных является производным от типа token. |

XML Schema подключается к XML-документу следующим образом:

```
<?xml version="1.0" encoding="Windows-1251"?>
<Journal xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="Z:\Язык XML\schemaschema.xsd">
  <Title>Телематика< /Title>
  <Article BookID="1" References="6">
    <Author FirstName="Андрей" MiddleName="Петрович"
LastName="Иванов" />
    <Title>
      XML-реляционные преобразования
    </Title>
    <Annotation>
```

В статье рассматривается задача построения реляционной модели на

основе XML-документов.

```
</Annotation>
<KeyWord>СУБД< /KeyWord>
<KeyWord>XML< /KeyWord>
</Article>
</Journal>
```

Пример реализации XSL-шаблона

Область действия шаблона определяется с помощью тега `<xsl:template>`. В значении атрибута `match` указывается наименование элемента XML-документа, к которому будет применяться правило оформления. Если элемент `xsl:template` не имеет атрибута `name`, атрибут `match` обязателен. В приведенном примере шаблон распространяется на все элементы документа. В стандартных тегах языка разметки указано название заголовка документа. После чего описан стиль для названия журнала. Затем в пределах `Journal` обрабатываются все статьи `Article` и значения записываются в таблицу. Выводится нумерация статей, и последовательно для элемента `Author` с помощью конструкции `". /@*"` перечисляются атрибуты. Конструкция `< xsl:text disable-output-escaping="yes"> < /xsl:text>` задает пробел. Таким же образом последовательно обрабатываются все необходимые элементы документа "Журнал".

```
<?xml version="1.0" encoding="Windows-1251"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <xsl:value-of select="//Title"/>
      </head>
      <body>
        <div style="font-size:24pt; font-style:bold">
          <xsl:value-of select="//Title"/>
        </div>
        <xsl:for-each select="Journal/Article">
          <table cellpadding="5">
            <tr>
              <td valign="top"> [<xsl:number
```

```

value="position()"/>]</td>
    <td>
        <xsl:for-each select="Author">
            <div style="font-size:14pt; font-
wieght:2000;">
                <xsl:value-of select="./@LastName"/>
                <xsl:text disable-output-
escaping="yes">&nbsp;< /xsl:text>
                <xsl:value-of select="./@FirstName"/>
                <xsl:text disable-output-
escaping="yes">&nbsp;< /xsl:text>
                <xsl:value-of select="./@MiddleName"/>
            </div>
        </xsl:for-each>
        <xsl:for-each select="Title">
            <div style="font-size:14pt; font-
style:bold; text-align:justify;">
                <xsl:value-of select="."/>
            </div>
        </xsl:for-each>
        <p> Аннотация:
            <xsl:for-each select="Annotation">
                <div style="color:000000; margin-
left:20px;">
                    <xsl:value-of select="."/>
                </div>
            </xsl:for-each>
        </p>
        <p>Ключевые слова:
            <xsl:for-each select="KeyWord">
                <div style="color:000000; margin-
left:20px;">
                    <xsl:value-of select="."/>
                </div>
            </xsl:for-each>
        </p>
    </td>

```



```

        </tr>
    </table>
</xsl:for-each>
</body>
</html>
</xsl:template>
</xsl:stylesheet>

```

Как видно из примера XSL-документ содержит некоторое множество инструкций, или правил [rules], по обработке XML. Данные инструкции - это элементы XML, выделенные в специальное пространство имен [name space]. Любая XSL-инструкция из данного пространства имен использует префикс "xsl:". Кроме XSL-инструкций, XSL-документ может содержать любые другие элементы, не определенные в пространстве имен XSL.

| Название инструкции | Описание |
|---------------------|---|
| xsl:apply-templates | В отсутствие атрибута select инструкция xsl:apply-templates обрабатывает все непосредственные потомки текущего узла, включая узлы текста. Чтобы обрабатывать не все непосредственные потомки, а лишь узлы, отобранные по некому выражению, используется атрибут select. |
| xsl:value-of | Используется для создания текстового узла в конечном дереве. Обязательный атрибут select выбирает значение создаваемого текстового узла. Тер <xsl:value-of select=строка запроса/> осуществляет подстановку результата. Из элемента Author с атрибутами LastName и FirstName создается параграф html, который содержит значение атрибута LastName из текущего узла, за которым следуют пробел и значение атрибута FirstName того же текущего узла. <pre> <xsl:template match="Author"> <p> <xsl:value-of select="@LastName"/> <xsl:text> </xsl:text> <xsl:value-of select="@FirstName"/> </p> </pre> |

| | |
|--------------|---|
| | </xsl:template> |
| xsl:number | <p>Используется для того, чтобы поместить в конечное дерево форматированное число. Подставляемое число может быть задано выражением. Атрибут value содержит выражение, которое обрабатывается, а полученный объект преобразуется в число, как при вызове функции number. Функция position используется при вычислении положения элемента. В тело элемента Journal подставляется порядковый номер статьи, указанный в квадратных скобках.</p> <pre><xsl:for-each select="Journal/Article"> [<xsl:number value="position()" />] </xsl: for-each></pre> |
| xsl:for-each | <p>Содержит шаблон, который обрабатывается для каждого узла, отобранного выражением, указанным в атрибуте select. Атрибут select является обязательным. Результатом обработки выражения является набор узлов. При обработке шаблона выбранный узел берется в качестве текущего узла, а весь список собранных узлов берется в качестве текущего набора узлов. Узлы обрабатываются в том порядке, в каком они следуют в документе, если нет указаний относительно сортировки.</p> <p>Тег <xsl:for-each> определяет область шаблона, на которую распространяется данный запрос. При обработке документа выбираются все значения текущего узла элементов.</p> <pre><xsl:for-each select="KeyWord"> <xsl:value-of select="." /> </xsl:for-each></pre> <p>При этом атрибут order-by позволяет упорядочить выводимые значения.</p> |
| xsl:if | <p>Инструкция обработки с условием xsl:if дает простую функциональность if-then. Имеет атрибут test, который определяет некое выражение. Содержимое элемента является шаблоном. Каждый второй ряд таблицы в документе "Журнал" раскрашивается желтым цветом.</p> <pre><xsl:template match="Journal"></pre> |

| | |
|------------|--|
| | <pre> <tr> <xsl:if test="position() mod 2 = 0"> <xsl:attribute name="bgcolor">yellow </xsl:if> <xsl:apply-templates/> </tr> </xsl:template> </pre> |
| xsl:choose | <p>Поддерживает выбор одного из нескольких возможных вариантов. Инструкция xsl:choose состоит из последовательности элементов xsl:when, за которой следует необязательный элемент xsl:otherwise. Каждый элемент xsl:when имеет единственный атрибут test, который задает некое выражение. Содержимое элементов xsl:when и xsl:otherwise является шаблоном. Если обрабатывается элемент xsl:choose, поочередно проверяются все элементы xsl:when. Из содержимого элемента Author выбираются через пробел значения атрибутов FirstName и MiddleName при наличии атрибута MiddleName, в противном случае, выводится сообщение "отсутствует".</p> <pre> <xsl:for-each select="Author"> <xsl:choose> <xsl:when test="./@MiddleName"> <xsl:value-of select="./@FirstName"/> <xsl:text disable-output- escaping="yes">&nbsp; <xsl:value-of select="./@MiddleName"/> </xsl:when> <xsl:otherwise> <xsl:text>отсутствует </xsl:otherwise> </xsl:choose> </xsl:for-each> </pre> |
| xsl:sort | <p>Сортировка задается с помощью инструкции xsl:sort, которая имеет атрибут select, значением которого является выражение. Указанное выражение вычисляется для каждого узла, подлежащего обработке. При этом данный узел используется в качестве текущего узла, а полный неупорядоченный список узлов, подлежащих обработке, выступает в</p> |

| | |
|--------------|---|
| | <p>роли текущего набора узлов.</p> <p>Сортировка представленных в документе "Журнал" заголовков статей, аннотаций и ключевых слов.</p> <pre> <xsl:template match="Journal"> <html> <head> <xsl:text>Журанл</xsl:text> </head> <body> <xsl:apply-templates select="Article"> <xsl:sort select="Title"/> <xsl:sort select="Annotation"/> <xsl:sort select="KeyWord"/> </xsl:apply-templates> </body> </html> </xsl:template> </pre> |
| xsl:text | <p>Символы фиксированных данных могут быть помещены в элемент xsl:text. Например, такое окружение может отменить режим удаления пробельных символов:</p> <pre> <xsl:textdisable-output-escaping = "yes" "no"> <!-- - Content: #PCDATA - -> </xsl:text> </pre> <p>Атрибут disable-output-escaping, может иметь значения yes или no (по умолчанию). Если значение атрибута yes, текстовый узел, полученный обработкой элемента xsl:text, должен быть представлен без маскирования.</p> |
| xsl:variable | <p>Используется для привязки переменных. Инструкцию xsl:variable можно ставить в любом месте шаблона, где допустимо использование инструкций. В этом случае, привязка видна всем последующим узлам, имеющим того же родителя и их потомков. Однако, привязка для самого элемента xsl:variable не видна.</p> <pre> <xsl:template name="Box"> <xsl:variable name="pencil" select="1"/> </xsl:template> </pre> |

| | |
|------------------------|--|
| <code>xsl:param</code> | <p>Данную инструкцию можно использовать как непосредственный потомок в начале элемента <code>xsl:template</code>. В данном контексте привязка переменной видна всем последующим узлам, имеющим того же родителя и их потомков. Для самого элемента <code>xsl:param</code> эта привязка не видна.</p> <p>Значение, указанное в переменной <code>xsl:param</code>, является значением по умолчанию. Если используется шаблон или стиль, в котором использован элемент <code>xsl:param</code>, то могут быть переданы параметры, которые будут использоваться вместо значений по умолчанию.</p> <pre><xsl:template name="Box"> <xsl:param name="pen" select="2"/> </xsl:template></pre> |
|------------------------|--|

Для выбора элементов для обработки и генерации текста XSL использует язык выражений, сформулированный в *XPath - языке адресации частей XML-документа*. Сокращенный синтаксис строки запроса, используемый при адресации:

| Синтаксис | Описание |
|------------------------------|---|
| <code>*</code> | Находит все дочерние элементы текущего узла документа. |
| <code>*/Article</code> | Находит все потомки во втором поколении <code>Article</code> текущего узла контекста. |
| <code>//Title</code> | Для выделения элементов, расположенных ниже по "дереву". |
| <code>//Journal/Title</code> | В документе, где располагается текущий узел, находит все элементы <code>Title</code> , имеющие родителем <code>Journal</code> . |
| <code>.</code> | Выделяет текущий узел документа. |
| <code>./Article</code> | Собирает элементы <code>Article</code> , являющиеся потомками текущего узла контекста. |
| <code>[]</code> | Заключает условие на значение в запросе. |
| <code>@*</code> | Выбор значения атрибута текущего узла документа. |

Чтобы просмотреть результат преобразования XML-документа в HTML формат при помощи созданного XSL-шаблона его необходимо подключить к исходному тексту XML-документа:

```
<?xml version="1.0" encoding="windows-1251"?>
<?xml-stylesheet type='text/xsl' href='template.xsl'?>
```

```

<!DOCTYPE Journal SYSTEM "Journal.dtd">
<Journal>
  <Title>Телематика</Title>
  <Article BookID="1" References="6">
    <Author FirstName="Андрей" MiddleName="Петрович"
LastName="Иванов"/>
    <Title>
      XML-реляционные преобразования
    </Title>
    <Annotation>
      В статье рассматривается задача построения реляционной модели на
основе XML-документов.
    </Annotation>
    <KeyWord>СУБД</KeyWord>
    <KeyWord>XML</KeyWord>
  </Article>
  <Article BookID="2" References="3">
    <Author FirstName="Михаил" MiddleName="Иванович"
LastName="Бодров"/>
    <Title>
      XSL-трансформации
    </Title>
    <Annotation>
      От XML-документов к HTML-документам.
    </Annotation>
    <KeyWord>HTML</KeyWord>
  </Article>
</Journal>

```