Санкт-Петербургский национальный
исследовательский университет информационных
технологий, механики и оптики

Центр авторизованного обучения
IT-технологиям

XML практикум

Санкт-Петербург
2020

Содержание

Введение в XML

XML (EXtensible Markup Language, расширяемый язык разметки) это язык разметки, который был создан для описания данных. В XML не заданных изначально тэгов, что позволяет разработчику самостоятельно определить свои собственные тэги и свою собственную структуру документа.

XML был создан для хранения, транспортировки и обмена данными; с его помощью можно реализовать обмен данными между различными системами.

XML обладает строгими синтаксическими правилами:

```
<?xml version="1.0"?>
<note date = “01-02-2012”>
        <to>Sunny</to>
        <from>Oliver</from>
        <Sbj>Hello</Sbj>
        <Message>This is a good day!</Message>
</note>
```

- Первая строка документа – XML-декларация;
- В документе обязательно должен присутствовать единственный корневой элемент;
- Все элементы должны иметь закрывающий тэг;
- Тэги являются регистрозависимыми;
- Необходимо строго соблюдать вложенной тэгов;
- Значения атрибутов всегда должны быть заключены в кавычки;
- XML сохраняет пробелы;
- В XML перевод строки всегда делается с помощью символа LF.

Наименования XML-элементов должны подчиняться следующим правилам:

- Названия могут содержать буквы, цифры и другие символы;
- Названия не могут начинаться с цифры или знака препинания;
- Названия не могут начинаться с букв xml;
- В названии не должно быть пробелов.

Элементы XML могут содержать в начальном тэге атрибуты, которые применяются для предоставления дополнительной информации об элементах. Значения атрибутов всегда должны быть заключены в кавычки, можно использовать и одинарные и двойные.

Поскольку имена элементов в XML не фиксированы, часто случаются конфликты, когда два различных документа используют одно и то же имя для описания двух различных типов элементов. **Пространства имен** XML дают возможность избегать конфликты имен элементов. Для этого атрибут **namespace** помещается в начальный тэг элемента с использованием следующего синтаксиса:

```
xmlns:namespace-prefix="namespace"
```

Далее используемый префикс используется в именах элементов, относящихся к данному пространству:

```
<?xml version="1.0"?>
<myns:note xmlns:myns = “http:///www.mynamespace.ru/myns”>
        <myns:to>Sunny</ myns:to>
        < myns:from>Oliver</ myns:from>
        < myns:Sbj>Hello</ myns:Sbj>
        < myns:Message>This is a good day!</ myns:Message>
</ myns:note>
```

Практическая работа №1: создание XML документов

Для выполнения данной и всех последующих практических работ вам потребуется среда **Notepad++,** а также плагин для данной среды **XML Tools**.

Для удобства работы в данном приложении Вы можете настроить русский язык интерфейса (Меню **Settings** – **Preferences** – вкладка **General**, меню **Localizations**), установить кодировку по умолчанию UTF-8: вкладка **Опции** (**Settings**) – **Настройки** (**Preferences**) – вкладка **Новый документ** (**New Document/Default Directory**) – меню **Кодировка** (**Encoding**), а также там же в меню **Синтаксис** (**Default Language**) установить **XML**.

Упражнение № 1: создание простого XML-документа

1. Откройте приложение Notepad ++ .
2. Напишите пролог для нового XML документа:

```
<?xml version = "1.0" encoding = "utf-8" ?>
```

3. Создайте XML-документ, описывающий данные кандидатов (резюме) на вакантную должность. Документ должен содержать поля со следующей информацией:

   - Id кандидата;
   - Имя;
   - Фамилия;
   - Отчество;
   - Текущее место работы;
   - Занимаемая в данный момент должность;
   - Дата рождения;
   - Образование;
   - Адрес;
   - Телефон;
   - Семейное положение;
   - Желаемый оклад в рублях.

Возможный вариант кода:

```
<?xml version = "1.0" encoding ="utf-8"?>
<resume>
        <candidate id = "a">
                <Name>Анна</Name>
                <LastName>Ошемкова</LastName>
```

```
            <PatronymicName>Петровна</PatronymicName>
            <Age>(самостоятельно напишите возраст на текущий момент)</Age>
            <WorkPlace>ООО "Проект"</WorkPlace>
            <Post>менеджер</Post>
            <DateOfBirth>12.09.1980</DateOfBirth>
            <Education>Высшее </Education>
            <Address>СПб, ул. Ленина 3-5</Address>
            <Phone>123-45-67</Phone>
            <FamilyStatus>замужем</FamilyStatus>
            <DesiredSalary currency = "RUR">30 000</DesiredSalary>
        </candidate>

        <candidate id = "b">
            <Name>Иван</Name>
            <LastName>Сидоренко</LastName>
            <PatronymicName>Иванович</PatronymicName>
            <Age>(самостоятельно напишите возраст на текущий момент)</Age>
            <WorkPlace>ООО "РусьСвет"</WorkPlace>
            <Post>экономист</Post>
            <DateOfBirth>02.03.1968</DateOfBirth>
            <Education>Высшее </Education>
            <Address>СПб, Серпуховская 13-9</Address>
            <Phone>987-65-54</Phone>
            <FamilyStatus>женат</FamilyStatus>
            <DesiredSalary currency = "RUR">50 000</DesiredSalary>
        </candidate>

        <candidate id = "c">
            <Name>Ольга</Name>
            <LastName>Гашекова</LastName>
            <PatronymicName>Сергеевна</PatronymicName>
            <Age>(самостоятельно напишите возраст на текущий момент)</Age>
            <WorkPlace>ООО "Строй"</WorkPlace>
            <Post>менеджер по продажам</Post>
            <DateOfBirth>19.12.1976</DateOfBirth>
            <Education>Среднее</Education>
            <Address>СПб, Дбуновская 12-4</Address>
            <Phone>122-33-44</Phone>
            <FamilyStatus>не замужем</FamilyStatus>
            <DesiredSalary currency = "RUR">35 000</DesiredSalary>
        </candidate>
</resume>
```

4. Сохраните документ под именем **resume.xml**.
5. Проверьте корректность синтаксиса только что созданного документа. Это можно сделать двумя способами: открыв документ в браузере или выбрав проверку синтаксиса в приложении Notepad++: меню **Плагины** (**Plugins**) – **XML Tools – Check XML syntax now** (см. рис. 1.1):

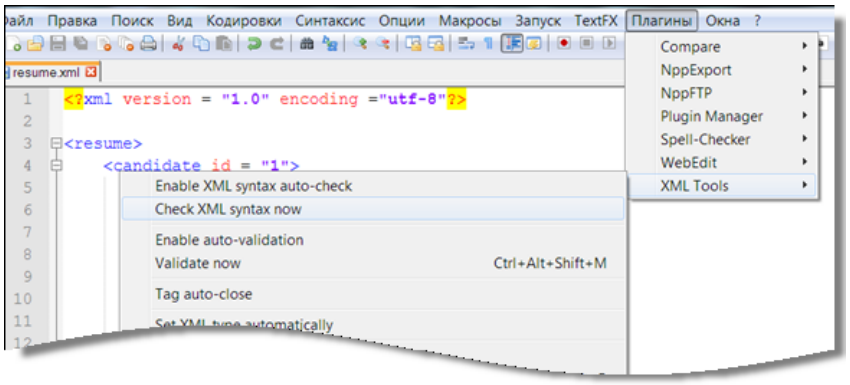

Рис. 1.1. Меню проверки синтаксиса документа в Notepad++

Упражнение №2: создание XML-документа

1. Откройте приложение Notepad ++ .
2. Создайте XML-документ, в котором самостоятельно опишите план (расписание) дисциплин процесса Вашего обучения (например, название предмета, фамилию преподавателя, время занятий, даты начала и окончания курсов и т.п.).
3. Сохраните файл под именем **lessons.xml** .
4. Проверьте правильность синтаксиса Вашего документа.

Упражнение №3: работа с пространствами имен

1. Измените ранее созданный документ **lessons.xml** таким образом, чтобы он использовал пространство имен http://www.itmo.ru/xml/lessons (с произвольным префиксом).

Например:

```
1  <?xml version = "1.0" encoding = "utf-8"?>
2  <myns:TimeTable xmlns:myns = "http://www.itmo.ru/xml/lessons">
3      <myns:item>
4          <myns:name>Курс №1</myns:name>
5          <myns:teacher>Преподаватель №1</myns:teacher>
6          <myns:begin>2012-01-01</myns:begin>
7          ...
```

Рис.1.2. Использование пространства имен

2. Сохраните документ под именем **lessons-ns.xml** и проверьте его правильность.
3. Допишите в документе **lessons-ns.xml** поле или поля, содержащие информацию об источниках литературы (и/или Internet-ресурсах) по каждому курсу.
4. Самостоятельно организуйте произвольное пространство имен для данных полей и задайте это пространство имен с помощью префикса в документе.
5. Проверьте корректность синтаксиса в документе.

DTD схемы

Задача определения типа документа (DTD - Document Type Definition) - задать допустимые сущности XML-документа. В нем задается структура документа и список допустимых элементов. DTD может быть объявлена внутри XML-документа или ссылкой на внешний файл DTD.

Если DTD включено в ваш исходный XML-документ, оно может быть заключено в декларацию DOCTYPE с помощью следующего синтаксиса:

```
<!DOCTYPE root-element [element-declarations]>
```

Пример XML-документа со встроенным DTD:

```
<?xml version="1.0"?>
	<!DOCTYPE note [
		<!ELEMENT note (to, from, Sbj, msg)>
		<!ELEMENT to      (#PCDATA)>
		<!ELEMENT from    (#PCDATA)>
		<!ELEMENT Sbj (#PCDATA)>
		<!ELEMENT msg  (#PCDATA)>
]>

<note>
	<to>Sunny</to>
	<from>Oliver</from>
	<Sbj>Hello</Sbj>
	<msg>This is a good day!</msg>
</note>
```

С точки зрения DTD, XML (и HTML) - документ состоит из следующих простых строительных блоков:

- Элементы (Elements);
- Тэги (Tags);
- Атрибуты (Attributes);
- Сущности (Entities)- это переменные, которые применяются для определения частей текста, который используется в нескольких местах. Ссылки на сущности вызывают эти части текста.
- PCDATA - парсируемые символьные данные;
- CDATA - непарсируемые символьные данные.

В DTD элементы XML объявляются с помощью DTD-декларации element. При этом используется следующий синтаксис:

```
<!ELEMENT element-name (element-content)>
```

Пустые элементы объявляются с помощью ключевого категориального слова EMPTY:

```
<!ELEMENT element-name EMPTY>
```

Элементы, содержащие только символьные данные объявляются с помощью #PCDATA в круглых скобках:

<!ELEMENT element-name (#PCDATA)>

Элементы, содержащие один или несколько дочерних элементов определяются перечислением имен дочерних элементов в круглых скобках:

<!ELEMENT element-name (child-element-name)>

Когда элементы объявляются с помощью последовательности, разделенной запятыми, дочерние элементы должны появляться в документе в той же последовательности. В полной DTD-декларации дочерние элементы должны также быть объявлены, при этом сами дочерние элементы также могут содержать свои дочерние элементы.

В DTD атрибуты объявляются с помощью DTD-декларации ATTLIST. При объявлении атрибутов используется следующий синтаксис:

<!ATTLIST element-name attribute-name attribute-type default-value>

Пример DTD:

<!ATTLIST payment type CDATA "check">

Типы значений атрибутов могут быть следующие:

Тип значения	**Значение атрибута**
CDATA	символьные данные
(en1|en2|..)	Список допустимых значений атрибута
ID	уникальный id
IDREF	id другого элемента
IDREFS	список других id
NMTOKEN	допустимое XML-имя
ENTITY	сущность
ENTITIES	список сущностей
NOTATION	запись (notation)
xml:	изначально заданное в XML значение

Описание применения атрибута может иметь следующие значения:

Значение	**Описание**
value	Атрибут имеет значение по умолчанию
#DEFAULT	Атрибут имеет значение по умолчанию
#REQUIRED	Атрибут обязательно должен присутствовать в элементе
#IMPLIED	Атрибут не обязательно должен присутствовать в элементе
#FIXED	Значение атрибута фиксировано

Сущности - это переменные, используемые для создания кратких ссылок на различные куски текста. Ссылки на сущности (entity references) - это указатели на сущности, являющиеся кусками какого-либо текста. Объявляемые сущности могут находиться как внутри, так и вне документа. Синтаксис:

```
<!ENTITY entity-name "entity-value">
```

Пример DTD:

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
```

Пример XML:

```
<author>&writer;&copyright;</author>
```

Практическая работа №2: работа с DTD схемами документов

Упражнение №1: создание DTD схемы документа

1. Откройте документ **lessons.xml**.
2. Реализуйте DTD схему для этого документа и сохраните схему в файле **lessons_DTD.dtd**.

Например, для xml-документа вида:

```
<?xml version="1.0" encoding="UTF-8"?>
<TimeTable>
	<item>
		<name>Курс №1</name>
		<teacher>Преподаватель №1</teacher>
		<begin>2012-01-01</begin>
		<end>2012-01-15</end>
		<description>Описание курса №1</description>
	</item>
	…
```

Может быть реализована DTD-схема вида:

```
<!ELEMENT TimeTable (item+)>
<!ELEMENT item (name, teacher, begin, end, description)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT teacher (#PCDATA)>
<!ELEMENT begin (#PCDATA)>
<!ELEMENT end (#PCDATA)>
<!ELEMENT description (#PCDATA)>
```

3. Подключите только что созданную DTD-схему к документу **lessons.xml**:

```
<!DOCTYPE TimeTable SYSTEM "lessons_DTD.dtd">
```

4. Проверьте соответствие документа **lessons.xml** созданной DTD-схеме (валидность документа).

Упражнение №2: работа с DTD-схемой

1. Дана следующая DTD-схема:

```
<!ELEMENT library (book_catalog, author_catalog)>

<!ELEMENT book_catalog (book*)>
<!ELEMENT book (authors?, title, publishing, annotation?)>
<!ELEMENT authors (author+)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT publishing (#PCDATA)>
<!ELEMENT annotation (#PCDATA)>

<!ELEMENT author_catalog (author_book*)>
<!ELEMENT author_book (#PCDATA)>

<!ATTLIST book
        id ID #REQUIRED
        isbn CDATA #IMPLIED
        year CDATA #REQUIRED
        issue (translation | original) "translation">

        <!ATTLIST author_book  idref IDREF #REQUIRED>

<!ENTITY linux "Linux">
<!ENTITY internet "Internet">
```

2. Прочитайте данную DTD-схему и создайте валидный для данной схемы xml-документ.
3. Подключите созданный xml-документ к DTD-схеме и проверьте корректность работы.

XML схемы (XML Schema)

XML-схема - это основанная на XML альтернатива DTD. XML-схема описывает структуру XML-документа.

```
<note>
        <to>Sunny</to>
        <from>Oliver</from>
        <Sbj>Hello</Sbj>
        <msg>This is a good day!</msg>
</note>
```

В данном примере элемент note является сложным элементом, поскольку он содержит другие элементы. Все другие элементы - простые, поскольку не содержат внутри других элементов. Синтаксис простых элементов :

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

Например:

```
<xs:element name="to" type="xs:string"/>
<xs:element name="from" type="xs:string"/>
<xs:element name="Sbj" type="xs:string"/>
<xs:element name="msg" type="xs:string"/>
```

Синтаксис сложных элементов:

```
<xs:element name="name">
   <xs:complexType>
      …
   </xs:complexType>
</xs:element>
```

Например:

```
<xs:element name="note">
   <xs:complexType>
     <xs:sequence>
         <xs:element name="to" type="xs:string"/>
         <xs:element name="from" type="xs:string"/>
         <xs:element name="Sbj" type="xs:string"/>
         <xs:element name="msg" type="xs:string"/>
     </xs:sequence>
   </xs:complexType>
</xs:element>
```

Ссылки на другие элементы;

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

<xs:element name="note">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="to"/>
      <xs:element ref="from"/>
      <xs:element ref="Sbj"/>
      <xs:element ref="msg"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

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

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

</xs:schema>
```

Практическая работа №3: работа с XML-схемами

Упражнение №1: создание простой XML-схемы

1. Откройте документ **lessons.xml**.
2. Удалите из документа указание на .dtd файл.
3. Реализуйте XML-схему для этого документа и сохраните схему в файле **lessons_XML.xml**.
4. Подключите созданную XML-схему к документу **lessons.xml** и проверьте соответствие документа **lessons.xml** созданной XML-схеме.

Упражнение №2: работа с простыми типами данных XML-схемы

1. Откройте документ **resume.xml**.
2. Реализуйте XML-схему для данного документа и сохраните схему в файле **resume_XML.xml.**
3. Подключите созданную XML-схему к документу **resume.xml** и проверьте его валидность.
4. Расширьте следующие простые типы данных в текущем в xml-документе так, чтобы:
   - Номер телефона выводился с помощью регулярного выражения и был вида ###-##-## (т.е. последовательность *«три символа дефис два символа дефис два символа»*);
   - Дата рождения могла быть не раньше 1 января 1947 года и не позже 1 января 1992 года;
   - Элемент «Семейное положение» ограничивался значениями «женат», «не женат», «замужем», «не замужем»;
   - Элемент «Образование» органичивался значениями «высшее», «среднее».

5. На базе простого типа данных «integer» создайте пользовательский тип данных «AgeType», ограничив его значениями от 20 до 65 лет и используйте этот тип для определения типа данных элемента с тэгом «Age».
6. Проверьте корректность работы схемы данных, сохраните изменения.

Упражнение №3: работа с комплексными типами данных XML-схемы

1. Скопируйте данные из документа **resume.xml** в новый файл и сохраните его под именем **resume_complex.xml.**
2. Измените новый документ таким образом, чтобы он сдержал отельные данные о резюме людей с высшим образованием и отдельные данные о резюме людей со средним образованием, например:

   ```
   <resume>
   ```

```
<HigherEducation>
        <Candidate>
                …

        </Candidate>
</HigherEducation>

<SecondaryEducation>
                …
</SecondaryEducation>
```

3. И у лиц с высшим образованием, и у лиц со средним образованием одни и те же элементы данных. Для того, чтобы в xml-документе их не описывать несколько раз, создайте XML-схему с произвольным именем, где реализуйте пользовательский тип данных, а затем опишите обе группы кандидатов на должность с помощью этого типа данных.
4. Проверьте корректность работы схемы данных и валидность xml-документа и сохраните изменения.