Федеральное государственное образовательное бюджетное учреждение
высшего образования
«Финансовый университет при Правительстве Российской Федерации»

Факультет информационных технологий и анализа больших данных

Департамент бизнес-информатики

# Пособие для самостоятельной работы студентов по дисциплине "Интеграция информационных систем на основе XML"

Рекомендуется для направления подготовки:

38.03.05 Бизнес-информатика

Профиль «ИТ-менеджмент в бизнесе»

Очная и заочная формы обучения

Квалификация (степень) выпускника: бакалавр

*Методическое обеспечение рассмотрено*
*и одобрено на заседании Департамента бизнес-информатики*
*(протокол № 10 от 28.04.2021)*
*Руководитель Департамента бизнес-информатики Н.Ф. Алтухова*

**Москва 2021**

Пособие для самостоятельной работы студентов составлено в соответствии с рабочей программой дисциплины «Интеграция информационных систем на основе XML» для студентов, обучающихся по профилю «ИТ-менеджмент в бизнесе» направления подготовки бакалавров 38.03.05 Бизнес-информатика.

**Разработчик:** Морозова О.А., канд. техн. наук, доцент Департамента бизнес-информатики.

Содержание

## ВВЕДЕНИЕ

Данный практикум предназначен для студентов, обучающихся по направлению подготовки 38.03.05 Бизнес-информатика (уровень бакалавриата) и самостоятельно выполняющих задания по дисциплине «Интеграция информационных систем на основе XML».

Практикум содержит описание порядка выполнения заданий для сомостоятельной работы, в нем также приводятся необходимые справочные данные, касающиеся стандартов языка определения XML-схем, правил записи регулярных выражений. Рассматривается сквозной пример, включающий этапы определения требований, разработки прототипа XML-документа, разработки и оптимизации XSD-схемы, создания модульной XSD-схемы.

Для выполнения заданий необходима MS Visual Studio или любой XML-редактор с функциями валидации документов и генерации XSD-схем.

## ГЛОССАРИЙ

| | |
|---|---|
| XML(eXtensible Markup Language) | Расширяемый язык разметки, предназначенный для хранения структурированных данных, обмена информацией между программами, а также для создания на его основе специализированных производных языков. |
| XSD(XML Schema): | Язык описания структуры документа. Описывает структуру данных, типы данных, значения данных. Используется для валидации XML-документов. |
| XML -парсер | Программный компонент, выполняющий разбор XML-документа и извлечение из него данных. |
| Пространство имен XML (XML Namespaces) | Способ уточнения (квалификации) имен элементов и атрибутов XML-документа. Каждое пространство имен – это коллекция имен, относящихся к определенной предметной области, в каждой все имена уникальны. Каждая коллекция должна иметь уникальный идентификатор (URI-адрес). Каждое XML-имя характеризуется идентификатором пространства имен и локальным именем в пределах своего пространства имен. Таким образом, появляется возможность определить элементы с одинаковыми именами, связанные с различными пространствами имен. |
| Валидные XML-документы | Документы, соответствующие ассоциированному с ним описанию документа (в контексте данного документа XSD-схеме). |
| Корректные (правильно сформированные) XML-документы | Документы, соответствующие только синтаксическим правилам оформления XML-документов. |
| Регулярные выражения | Язык описания некоторого множества строк с помощью шаблонов, без необходимости перечисления всех элементов этого множества. |

# 1 ОБЩИЕ СВЕДЕНИЯ

XML –это язык разметки документов, предназначенный для хранения структурированных данных, обмена информацией между программами, а также для создания на его основе специализированных производных языков. Язык поддерживается Консорциумом Всемирной паутиныW3C, на сайте которого можно найти официальную спецификацию языка[1].

Расширяемый язык разметки XML (eXtensible Markup Language) начал широко использоваться с конца 90-х годов прошлого века для переноса данных между различными информационными системами и описания бизнес-транзакций.

В начале 2000-х годов, когда на первый план вышли проблемы интеграции разнородных приложений, появилась концепция веб-служб и основанная на ней парадигма сервис-ориентированной архитектуры, начали развиваться технологии управления бизнес-процессами (BPM). Крупнейшие поставщики программного обеспечения активно разрабатывали внутрикорпоративные стандарты описания, реализации и исполнения бизнес-процессов для своих программных систем. Многочисленные спецификации языков описания бизнес-процессов (JPDL, XLang, WSFL, WSCL, BPSS, WSCI) разрабатывались конкурирующими вендорами. Все эти языки формировались на основе языка XML. На сегодняшний день общепризнанными стандартами процессного управления являются основанные на XML языки: WSDL (Web Service Definition Language), WS-BPEL (Web Services Business Prosecc Execution Language) и WS-CDL (Web Services Choreography Discription Language).

---

[1] Extensible Markup Language (XML) 1.0 (Fifth Edition) https://www.w3.org/TR/xml/#entproc

Целесообразность использования языка XML, и основанного на нем технологий, для решения интеграционных задач обусловлена следующими преимуществами:

XML является самоописывающим языком. XML-документ содержит элементы и атрибуты, использование которых подчинено строгим правилам, однако имена элементов и атрибутов имеют содержательный характер. Такой документ может быть легко прочитан прикладной программой (синтаксическим анализатором) и доступен для прочтения человеку. Имеется возможность задавать имена в соответствии с принятыми в конкретной прикладной области стандартами, а не на основе жесткого синтаксиса.

1. XML- самодокументируемый формат, предназначенный не только для описания данных, но и для определения метаданных, в том числе для описания дополнительных правил, ограничивающих данные в соответствии с информационной моделью предметной области. Например, XSD схема позволяет задать следующие типы ограничений:

   Базовые и производные типы данных (строки, даты, логический и др.);

   - Расширенные типы данных (произвольные пользовательские типы данных);
   - Фасеты (дополнительные ограничения, такие как длина, дробные числа и т.д.);
   - Границы значений (наибольшее и наименьшее значение);
   - Перечисление (набор допустимых значений для атрибута);
   - Условия кардинальности (вхождение повторяющегося элемента данных);
   - Шаблоны.

2. XML является универсальным языком, подходящим для описания практически любых типов документов. В формате XML могут быть

описаны основные структуры данных - такие как записи, списки и деревья.

3. XML является расширяемым языком. Любую xml-структуру можно разрабатывать в расчете на оперативное сокращение или расширение. Расширение возможно как за счет включения в структуру новых элементов и\или атрибутов, так и путем использования механизмов пространств имен.
4. Обеспечивает межплатформенное взаимодействие. Поскольку XML -это текстовый формат, требования для организации взаимодействия между отправляющей и принимающей платформами минимальны и сводятся к двум моментам:
   - возможность читать и создавать текстовые файлы в кодировке Юникод (UTF-8 или ASCII),
   - наличие синтаксического анализатора для обработки xml-документов.
5. Безусловным преимуществом является наличие международных стандартов (поддерживаются консорциумом W3C www.w3.org). Язык имеет строго определенный синтаксис и требования к анализу, что позволяет ему оставаться простым, эффективным и непротиворечивым. Актуальными спецификациями являются XML 1.0, XML Information Set, Namespaces in XML, XML Schema. В настоящее время XML широко используется для хранения и обработки документов, поддерживается всеми ведущими производителями ПО.

XML ориентирован на повторное использование, что позволяет уменьшить стоимость разработки интеграционного решения, снизить эксплуатационные затраты, а также способствует продвижению корпоративных и отраслевых стандартов. Например, XML-схемы могут быть спроектированы как модульные внешние компоненты. Нестандартные

элементы и типы данных целесообразно определять в отдельных XML-схемах, а затем многократно использовать посредством их включения в документы или с помощью ссылки на них.Основным инструментальным средством для построения математической модели проекта является MS Project, а также MS Word, MS Excel, MS Visio, BizAgi Process Modeler.

Несмотря на то, что для обмена данными набирает популярность формат Json, (в силу своей более компактной структуры и простоты обработки файлов), технологии интеграции, основанные на XML остаются базовыми технологиями в силу высокой степени стандартизации и наличию возможностей, отсутствующих у Json.

# 2 ОПРЕДЕЛЕНИЕ ТРЕБОВАНИЙ И РАЗРАБОТКА ПРОТОТИПА XML-ДОКУМЕНТА

**Цель работы:** определить требования к данным и создать прототипы xml-файлов, используемых при информационном взаимодействии между корпоративными приложениями.

**Порядок выполнения работы**:

1. Выбрать вариант (согласно Приложению 1).
2. Описать требования к структуре и формату передаваемых данных.

Описание приводится в табличной форме. Файл должен содержать не менее 20 элементов и атрибутов.

Таблица 1

| Код элемента | Содержание элемента | Тип | Формат | Описание | Дополнительная информация |
|---|---|---|---|---|---|
| | | | | | |

В графе "Код элемента" записывается условное обозначение элемента сообщения.

В графе "Содержание элемента" записывается условное обозначение атрибута или дочернего элемента, входящего в состав элемента.

В графе "Тип" записывается один из символов О, Н, ОА, НА, У, П, М. Символы имеют следующий смысл:

- О - обязательный элемент;
- Н - необязательный элемент;
- ОА – обязательный атрибут;
- НА – необязательный атрибут;
- У - условно-обязательный элемент;
- П - предписанный элемент;

- М - элемент, определяющий множественность данных, может добавляться к указанным выше символам.

**Обязательный элемент** - это элемент, который должен обязательно присутствовать в файле.

**Необязательный элемент** - это элемент, который может как присутствовать, так и отсутствовать в файле.

**Обязательный атрибут** – это атрибут, который должен обязательно присутствовать в элементе.

**Необязательный атрибут** - это атрибут, который может как присутствовать, так и отсутствовать в элементе.

**Условно-обязательный элемент** - это элемент, присутствие которого в файле обусловлено значениями, наличием или отсутствием других элементов этого же файла. В случае выполнения условия присутствия (УП) условно-обязательный элемент по всем своим свойствам приравнивается к обязательному, а в случае невыполнения - к необязательному.

**Предписанный элемент** - это элемент, код которого должен обязательно присутствовать в файле, в то время как значения может и не быть.

Единичные элементы - это показатели, которые встречаются в сообщении один раз. **Множественные элементы** - это показатели таких частей сообщения, которые содержат заранее неизвестное число однотипных строк таблицы (табличные множественные показатели) или однотипных фрагментов формы иной структуры.

В графе "Формат" для каждого атрибута указывается, - символ формата, а вслед за ним в круглых скобках - максимальная длина атрибута. Если круглых скобок нет - то длина атрибута произвольна.

Символы формата соответствуют вышеописанным обозначениям:

- Т - <текст>;

- N - <число>;
- D - <дата>;
- К - <код>;
- S - <элемент>; составной элемент, описывается отдельно;
- SA - <элемент>; составной элемент, содержащий атрибут, описывается после описания основного элемента;
- B –<булево выражение>;
- E - <пустое выражение>

Если значением атрибута является дробное десятичное число, то в графе "Формат" указывается формат его представления в виде N(m.k), где m - максимальное количество знаков в числе, включая целую и дробную часть числа, десятичную точку и знак "-" (минус), а k - число знаков дробной части числа.

В графе "Наименование" указывается наименование элемента или атрибута.

Если элемент или атрибут имеет в рамках данного формата ограниченное количество возможных значений, то в графе "Дополнительная информация" указывается список этих значений.

Пример заполнения таблицы (данные о контрагентах). В данном примере описана структура XML-документа, содержащего только элементы.

Таблица 2

| Код элемента | Содержание элемента | Тип | Формат | Описание | Дополнительная информация |
|---|---|---|---|---|---|
| Контрагенты | Контрагент | O | S | | |

| | | | | | |
|---|---|---|---|---|---|
| Контрагент | Код<br>Наименование<br>ИНН<br>КПП<br>Адрес<br>КонтактноеЛицо | OM | S | | |
| Код | | O | K | Уникальный код контрагента | 4х-значный буквенно-цифровой код, 1-ый символ бука Ф или Ю для физических или юридических лиц |
| Наименование | | O | T | Наименование контрагента | Строка не более 100 символов |
| ИНН | | O | T | ИНН контрагента | 10и-значный цифровой код |
| КПП | | H | T | КПП контрагента | 9и-значный цифровой код |
| Адрес | Индекс<br>Область<br>Район<br>Город<br>НаселенныйПункт<br>Улица<br>Дом | O | S | Блок адреса контрагента | |
| КонтактноеЛицо | ФИО<br>Телефон | OM | S | Контактные данные | |
| Индекс | | O | N | | 6и-значный цифровой код |
| Область | | П | T | | Возможные значения:<br>• Московская<br>• Ленинградская<br>• Владимирская<br>• Тверская |
| Район | | П | T | | Строка не более 30 символов |
| Город | | П | T | | Строка не более 20 символов |
| НаселенныйПункт | | П | T | | Строка не более 30 символов |

| | | | | | |
|---|---|---|---|---|---|
| Улица | | П | Т | | Строка не более 20 символов |
| Дом | | П | Т | | |
| ФИО | | О | Т | ФИО контактного лица | Строка не более 100 символов |
| Телефон | СлужебныйТелефон<br>МобильныйТелефон | О | S | Блок телефонов контактного лица | |
| СлужебныйТелефон | | Н | Т | Рабочий телефон | Заполняется по маске<br>+7(***)*******<br>или<br>+7 (***) *** ** ** |
| МобильныйТелефон | | О | Т | Мобильный телефон | Заполняется по маске<br>+7(***)*******<br>или<br>+7 (***) *** ** **<br>Допускается от 1 до 3-х номеров |

3. Создать прототип xml-документа, для этого:

- Открыть MS Visual Studio
- Выполнить команду File / New / File
- В окне «New file» выбрать шаблон XML file

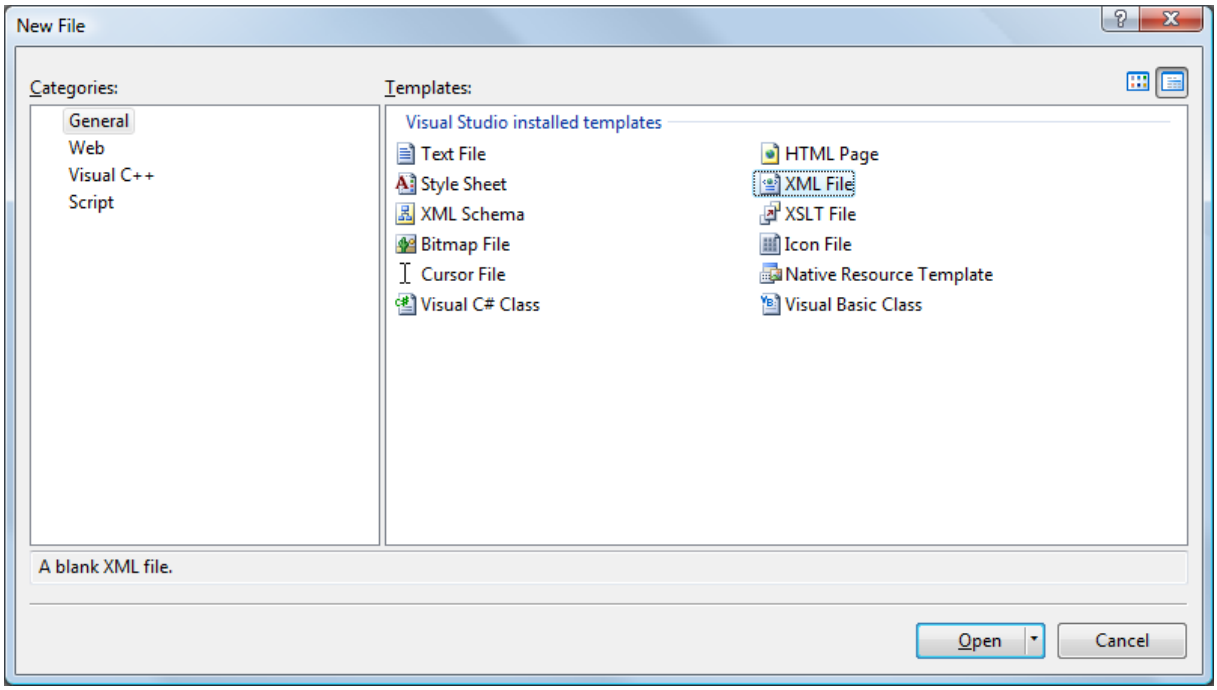


Рис. 1. Создание xml-файла в Visual Studio

- Ввести xml-код документа. Заполнить прототип документа данными в соответствии с заданными ограничениями. Требования к синтаксической корректности XML-документа представлены в ПРИЛОЖЕНИИ 2.
- Сохранить документ под именем ***XMLFile1.xml***.

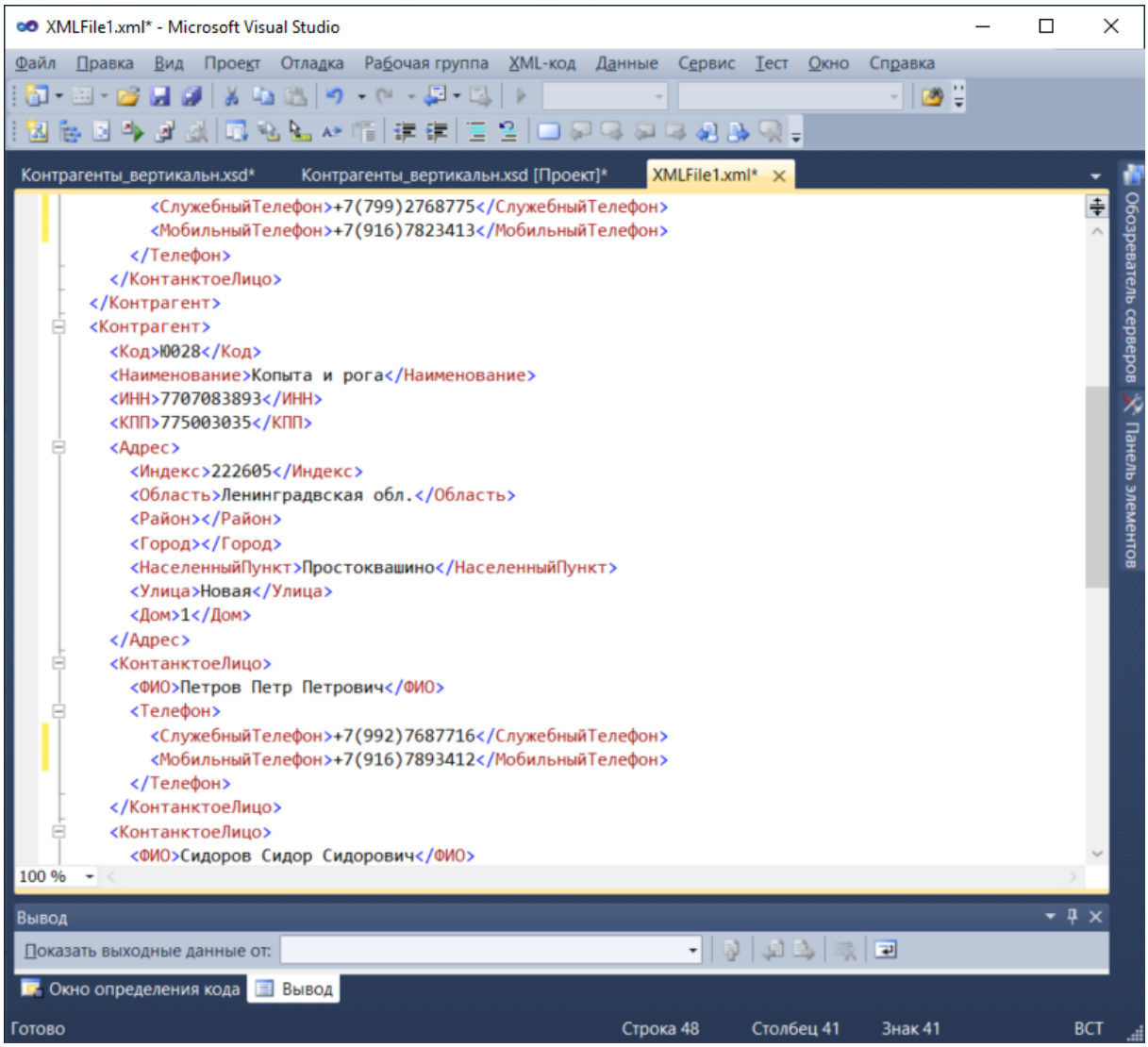


Рис. 2. Создание xml-файла в MS Visual Studio

Пример созданного прототипа XML-документа приводится ниже:

```
<?xml version="1.0" encoding="windows-1251"?>
<Контрагенты xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="Контрагенты_вертикальн.xsd">
  <Контрагент>
    <Код>Ю023</Код>
    <Наименование>Рога и копыта</Наименование>
    <ИНН>1232345678</ИНН>
    <КПП>775003657</КПП>
    <Адрес>
      <Индекс>118200</Индекс>
      <Область></Область>
      <Район></Район>
```

```
      <Город>Москва</Город>
      <НаселенныйПункт></НаселенныйПункт>
      <Улица>Ленинградский проспект</Улица>
      <Дом>100</Дом>
    </Адрес>
    <КонтанктоеЛицо>
      <ФИО>Иванов Иван Иванович</ФИО>
      <Телефон>
        <СлужебныйТелефон>+7(799)2768775</СлужебныйТелефон>
        <МобильныйТелефон>+7(916)7823413</МобильныйТелефон>
      </Телефон>
    </КонтанктоеЛицо>
  </Контрагент>
  <Контрагент>
    <Код>Ю028</Код>
    <Наименование>Копыта и рога</Наименование>
    <ИНН>7707083893</ИНН>
    <КПП>775003035</КПП>
    <Адрес>
      <Индекс>222605</Индекс>
      <Область>Ленинградвская обл.</Область>
      <Район></Район>
      <Город></Город>
      <НаселенныйПункт>Простоквашино</НаселенныйПункт>
      <Улица>Новая</Улица>
      <Дом>1</Дом>
    </Адрес>
    <КонтанктоеЛицо>
      <ФИО>Петров Петр Петрович</ФИО>
      <Телефон>
        <СлужебныйТелефон>+7(992)7687776</СлужебныйТелефон>
        <МобильныйТелефон>+7(916)7893412</МобильныйТелефон>
      </Телефон>
    </КонтанктоеЛицо>
    <КонтанктоеЛицо>
      <ФИО>Сидоров Сидор Сидорович</ФИО>
      <Телефон>
        <СлужебныйТелефон>+7(799)2768777</СлужебныйТелефон>
        <МобильныйТелефон>+7(916)7893413</МобильныйТелефон>
      </Телефон>
```

```
        </КонтанктоеЛицо>
      </Контрагент>
    </Контрагенты>
```

4. Предложить другую структуру документа, описать структуру и разработать прототип документа, см. пример.

В данном примере представлена структура XML-документа, содержащего элементы и атрибуты.

Таблица 3

| Код элемента | Содержание элемента | Тип | Формат | Описание | Дополнительная информация |
|---|---|---|---|---|---|
| Контрагенты | Контрагент | O | S | | |
| Контрагент | Код<br>Наименование<br>ИНН<br>КПП<br>Адрес<br>КонтактноеЛицо | OM | SA | | |
| Код | | OA | K | Уникальный код контрагента | 4х-значный буквенно-цифровой код, 1-ый символ бука Ф или Ю для физических или юридических лиц |
| Наименование | | O | T | Наименование контрагента | Строка не более 100 символов |
| ИНН | | O | T | ИНН контрагента | 10и-значный цифровой код |
| КПП | | H | T | КПП контрагента | 9и-значный цифровой код |
| Адрес | Индекс<br>Область<br>Район<br>Город<br>НаселенныйПункт<br>Улица<br>Дом | O | SA | Блок адреса контрагента | |

| КонтактноеЛицо | ФИО<br>Телефон | OM | SA | Контактные данные | |
|---|---|---|---|---|---|
| Индекс | | HA | N | | 6и-значный цифровой код |
| Область | | HA | T | | Возможные значения:<br>• Московская<br>• Ленинградская<br>• Владимирская<br>• Тверская |
| Район | | HA | T | | Строка не более 30 символов |
| НаселенныйПункт | | HA | T | | Строка не более 30 символов |
| Улица | | HA | T | | Строка не более 20 символов |
| Дом | | HA | T | | |
| ФИО | | OA | T | ФИО контактного лица | Строка не более 100 символов |
| Телефон | СлужебныйТелефон<br>МобильныйТелефон | O | S | Блок телефонов контактного лица | |
| СлужебныйТелефон | | H | T | Рабочий телефон | Заполняется по маске<br>+7(***)******* или<br>+7 (***) ***-**-** |
| МобильныйТелефон | | O | T | Мобильный телефон | Заполняется по маске<br>+7(***)******* или<br>+7 (***) ***-**-**<br>Допускается от одного до трех номеров. |

Пример прототипа XML-документа, содержащего атрибуты (документ получается короче).

```
<?xml version="1.0" encoding="windows-1251"?>
<Контрагенты>
      <Контрагент Код="Ю023">
            <Наименование>Рога и копыта</Наименование>
            <ИНН>1232345678</ИНН>
            <КПП>775003657</КПП>
```

```
            <Адрес    Индекс="118200"    Город="Москва"    НаселенныйПункт=""
Улица="Ленинградский проспект" Дом="100"/>
            <КонтанктоеЛицо ФИО="Иванов Иван Иванович">
                  <Телефон>
                     <СлужебныйТелефон>+7(495)2113477</СлужебныйТелефон>
                     <МобильныйТелефон>+7(905)67845234</МобильныйТелефон>
                  </Телефон>
            </КонтанктоеЛицо>
      </Контрагент>
      <Контрагент Код="Ю028">
            <Наименование>Копыта и рога</Наименование>
            <ИНН>7707083893</ИНН>
            <КПП>775003035</КПП>
            <Адрес  Индекс="222605"  Область="Ленинградская  обл."  Район=""
      НаселенныйПункт="Простоквашино" Улица="Новая" Дом="1"/>
            <КонтанктоеЛицо ФИО="Петров Петр Петрович">
                  <Телефон>
                        <МобильныйТелефон>+7(916)7893413</МобильныйТелефон>
                  </Телефон>
            </КонтанктоеЛицо>
            </Контрагент>
      </Контрагенты>
```

# 3 РАЗРАБОТКА ПРОСТОЙ XSD-СХЕМЫ

**Цель работы:** разработать простую XSD-схему для проверки прототипа XML-документа на соответствие требованиям.

**Порядок выполнения работы**:

1. Сгенерировать XSD-схему на основании разработанного выше прототипа документа

   Для этого:

   - Открыть файл ***XMLFile1.xml***, полученный в ходе выполнения предыдущего задания, в MS Visual Studio.
   - Выполнить команду**XML-код / Создать схему.**

   В результате будет сгенерирована XSD-схема.

   Пример кода автоматически сгенерированной схемы приведен ниже.

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema attributeFormDefault="unqualified" elementFormDefault="qualified
" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="xs:string" />
              <xs:element name="Наименование" type="xs:string" />
              <xs:element name="ИНН" type="xs:unsignedLong" />
              <xs:element name="КПП" type="xs:unsignedInt" />
              <xs:element name="Адрес">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Индекс" type="xs:unsignedInt" />
                    <xs:element name="Область" type="xs:string" />
                    <xs:element name="Район" />
                    <xs:element name="Город" type="xs:string" />
                    <xs:element name="НаселенныйПункт" type="xs:string" />
                    <xs:element name="Улица" type="xs:string" />
                    <xs:element name="Дом" type="xs:unsignedByte" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element name="КонтанктоеЛицо">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ФИО" type="xs:string" />
                    <xs:element name="Телефон">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="СлужебныйТелефон" type="xs:unsi
gnedLong" />
                          <xs:element name="МобильныйТелефон" type="xs:unsi
gnedLong" />
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
```

```
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

2. Сохранить схему в рабочем каталоге (в одном каталоге с xml-файлом). Файл схемы будет иметь расширение xsd.

   Пусть в нашем примере это будет файл XMLFile1.xsd

3. Доработать XSD- схему документа, добавив в нее фасеты, задающие ограничения на значения в соответствии с заданными ограничениями на значения данных.

   Например, задаем ограничение на значение кода контрагента

```
<xs:element name="Код">
                <xs:simpleType>
                 <xs:restriction base="xs:string">
                  <xs:length fixed="true" value="4"/>
                  <xs:pattern value="[Ю|Ф][0-9]{3}"/>
                 </xs:restriction>
                </xs:simpleType>
       </xs:element>
```

   В Приложениях 3 и 4 приводится краткая справочная информация по элементам XSD-схемы. В Приложении 5 содержится описание элементов языка регулярных выражений.

   Для быстрого перемещения по иерархии узлов XSD-схемы в MS Visual-studio используется представление Обозреватель xml-схемы (команда Вид / Обозреватель xml-схемы). Отображается подчиненность элементов схемы, типы данных элементов и их кардинальность.


Рис. 3. Обозреватель xml-схемы

Ограничения должны быть прописаны в XSD-схеме для всех требований к допустимым значениям данных.

4. Связать XML-документ с XSD- схемой.

   Для этого в корневой тег xml-документа добавить ссылку на схему

```
<?xml version="1.0" encoding="windows-1251"?>
<Контрагенты xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="XMLFile1.xsd">
```

   Редактор XML MS Visual Studio осуществляет проверку открытого XML-документа на соответствие ассоциированной с ним XSD-схеме. Строки, в которых обнаружено несоответствие подчеркиваются, при наведении курсора на строку выводится описание ошибки.

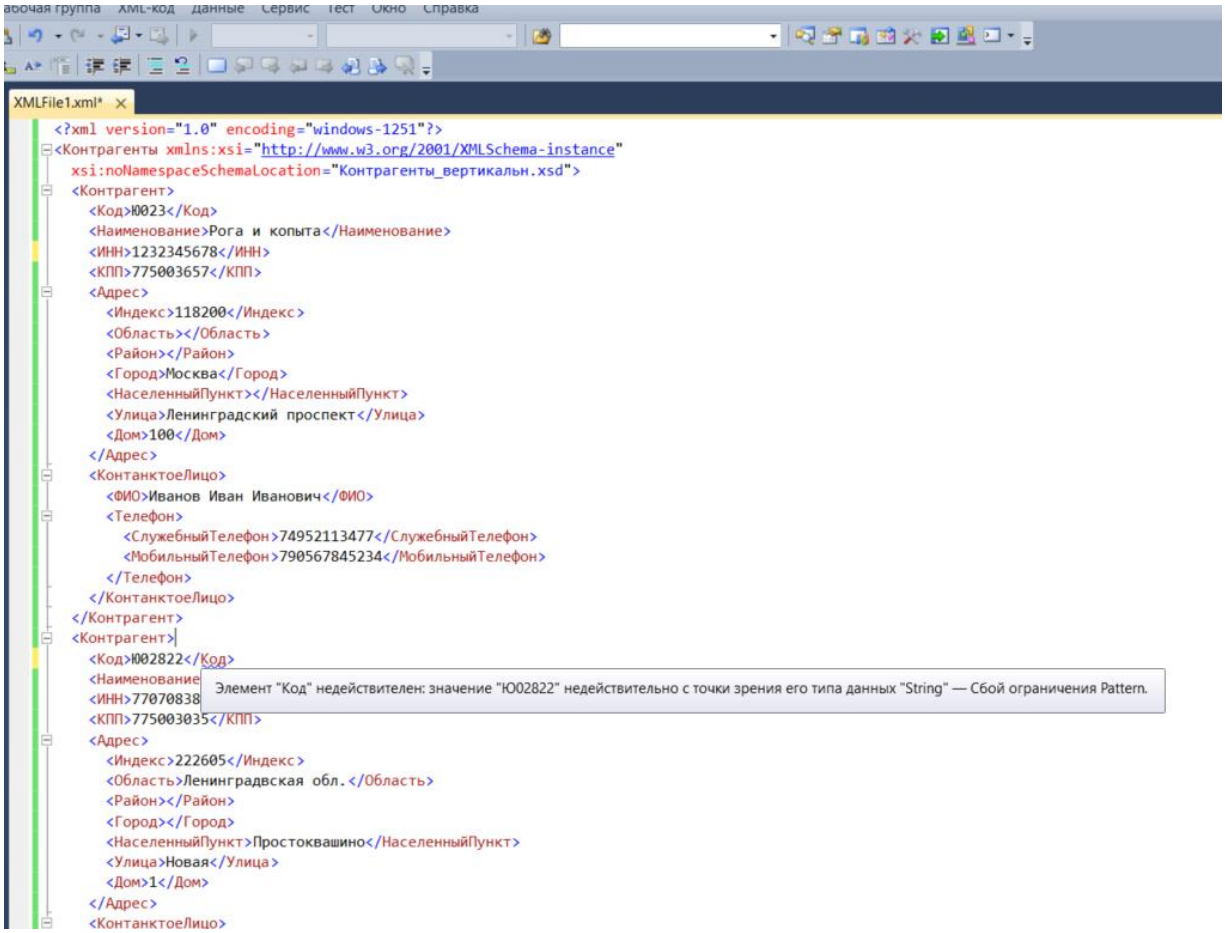


Рис. 4. Проверка валидности xml-файла в MS Visual Studio

5. Повторить операции 1-4 для второго прототипа xml-документа, содержащего атрибуты.

# 4 СОЗДАНИЕ ПРОСТЫХ ПОЛЬЗОВАТЕЛЬСКИХ ТИПОВ ДАННЫХ

**Цель работы:** разработать простые пользовательские типы данных, предназначенные для повторного использования в рамках одной XSD-схемы.

**Порядок выполнения работы**:

Создать пользовательские простые типы данных для всех ограничений на данные. Для этого:

1. Описываем в схеме пространство имен для пользовательских типов данных. Пусть это будет пространство имен с URI http://MyNames.com.

   Добавляем в корневой тег xml-схемы атрибут xmlns="http://MyNames.com”.
2. Чтобы иметь возможность использовать имена нового пространства имен без префикса, делаем новое пространство имен пространством имен по умолчанию. Для этого добавляем в корневой тег схемы атрибут targetNamespace=”http://MyNames.org”.
3. Чтобы использовать в коде схемы комментарии на русском языке добавляем в корневой тег схемы атрибут xml:lang="ru"

   Корневой тег схемы будет выглядеть следующим образом:

```
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com"
targetNamespace="http://MyNames.com" xml:lang="ru">
```

4. Создаем пользовательский простой тип данных с именем Код_контр

```
  <xs:simpleType name="Код_контр">
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="4"/>
      <xs:pattern value="[Ю|Ф][0-9]{3}"/>
    </xs:restriction>
  </xs:simpleType>
```

Описание пользовательского простого типа можно расположить в начале или в конце XSD-схемы.

5. Ссылаемся на созданный пользовательский тип данных при описании элемента.

Редактор XML предлагает полную проверку схемы по мере ввода, цветовое кодирование и технологию IntelliSense. Технология IntelliSense выводит список разрешенных элементов и атрибутов, из которого нужно

выбрать необходимое значение. Созданный пользовательский тип данных должен оказаться в списке доступных типов данных.

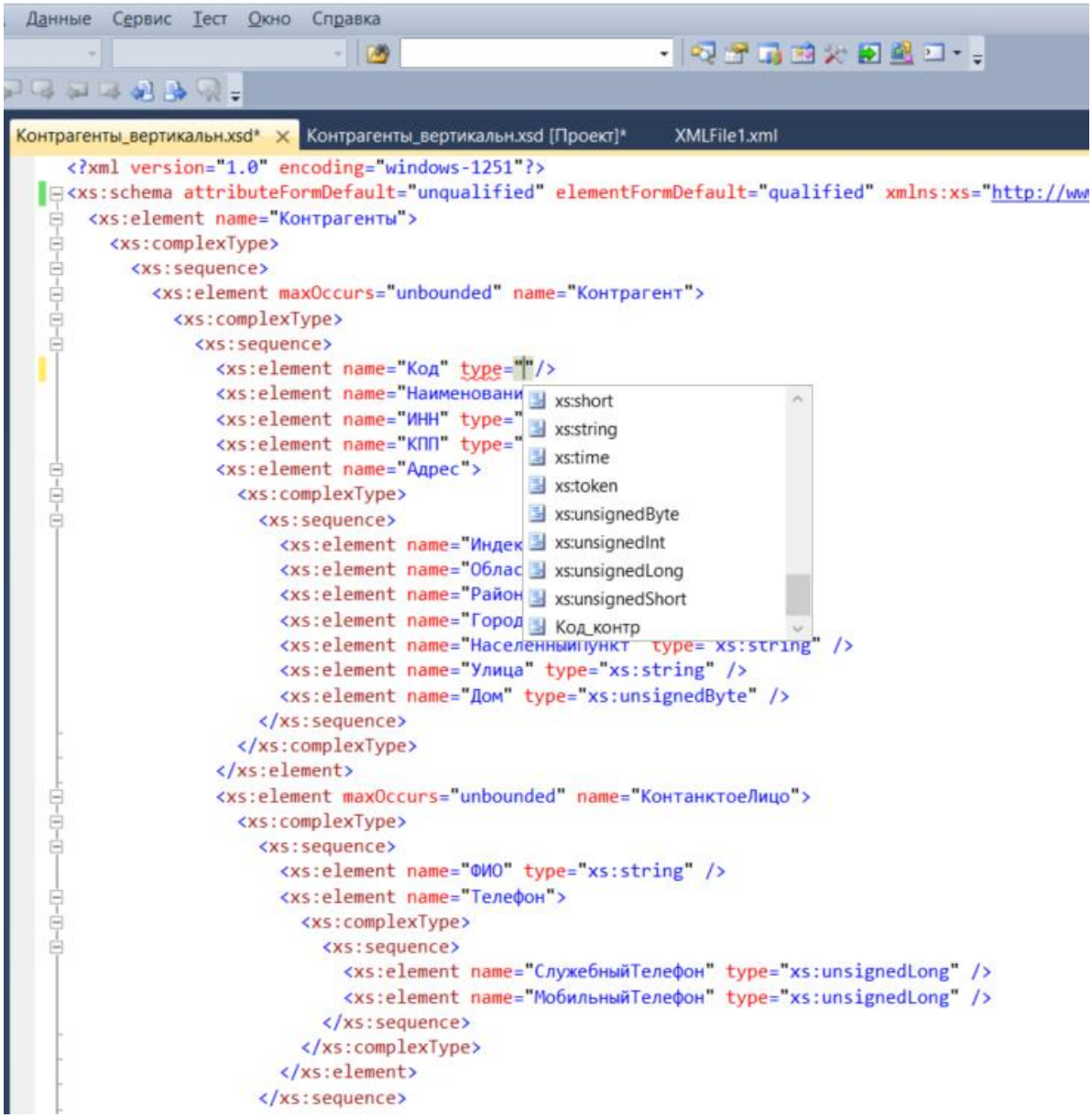


Рис. 5. Технология IntelliSense при выборе типа данных

6. Действуя аналогично (пп.4-5) создаем пользовательские простые типы данных для всех остальных ограничений на данные.
7. Для облегчения последующей работы со схемой добавляем комментарии в каждый новый простой тип данных.
8. Добавляем в схему условия обязательности\необязательности элементов и условия кардинальности.

Итоговый код схемы будет выглядеть следующим образом:

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://MyNames.com"
targetNamespace="http://MyNames.com" xml:lang="ru">
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="Строка100" />
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
              <xs:element name="Адрес">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="Индекс" type="Индекс"/>
                    <xs:element name="Область" type="Область"/>
                    <xs:element name="Район"  type="Строка30"/>
                    <xs:element name="Город" type="Строка20"/>
                    <xs:element name="НаселенныйПункт" type="Строка30" />
                    <xs:element name="Улица" type="Строка20"/>
                    <xs:element name="Дом" type="xs:unsignedByte" />
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
              <xs:element maxOccurs="unbounded" name="КонтанктоеЛицо">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ФИО" type="Строка100" />
                    <xs:element name="Телефон">
                      <xs:complexType>
                        <xs:sequence>
                          <xs:element name="СлужебныйТелефон"
type="телефон"  minOccurs="0" maxOccurs="1"/>
```

```
                          <xs:element name="МобильныйТелефон"
type="телефон"  minOccurs="1" maxOccurs="3"/>
                        </xs:sequence>
                      </xs:complexType>
                    </xs:element>
                  </xs:sequence>
                </xs:complexType>
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:annotation>
    <xs:documentation>Блок описания простых типов</xs:documentation>
  </xs:annotation>
   <xs:simpleType name="Код_контр">
    <xs:annotation>
      <xs:documentation>Простой тип для кода контрагент</xs:documentation>
     </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="4"/>
      <xs:pattern value="[Ю|Ф][0-9]{3}"/>
    </xs:restriction>
  </xs:simpleType>
   <xs:simpleType name="ИНН">
    <xs:annotation>
      <xs:documentation>Простой тип для ИНН</xs:documentation>
     </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="10"/>
      <xs:pattern value="\d{10}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="КПП">
    <xs:annotation>
      <xs:documentation>Простой тип для КПП</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
```

```
      <xs:length fixed="true" value="9"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="телефон">
    <xs:annotation>
      <xs:documentation>Простой тип для номера телефона</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="(\+7|8)[\s(]*\d{3}[)\s]*\d{3}[\s-]?\d{2}[\s-
]?\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Индекс">
    <xs:annotation>
      <xs:documentation>Простой тип для индекса в блоке
адреса</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="6"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Область">
    <xs:annotation>
      <xs:documentation>Справочник областей</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Московская"/>
      <xs:enumeration value="Ленинградская"/>
      <xs:enumeration value="Владимирская"/>
      <xs:enumeration value="Тверская"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка30">
    <xs:annotation>
      <xs:documentation>Строка не более 30 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
```

```
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка20">
    <xs:annotation>
      <xs:documentation>Строка не более 20 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка100">
    <xs:annotation>
      <xs:documentation>Строка не более 100 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

В обозревателе xml-схемы будет показана структура схемы с учетом созданных простых типов, эти типы являются глобальными, областью действия для них является данная схема (см. Рис. 6). Глобальные элементы схемы в обозревателе находятся на одном уровне иерархии с корневым элементом <Контрагенты>.

Выполните команду меню Вид/Конструктов, MS Visual Studio вернется к стартовому представлению XSD-схемы (Рис. 7). Перейдем к представлению для просмотра модели содержимого и перетащим в рабочую область корневой тег XSD-схемы <Контрагенты>. Представление модели содержимого предоставляет графическое представление сведений о локальных и глобальных узлах схемы и их компонентах, включая простые

и сложные типы, элементы, группы моделей, атрибуты и группы атрибутов. XML-комментарии и инструкции по обработке нельзя просматривать в представлении модели содержимого.

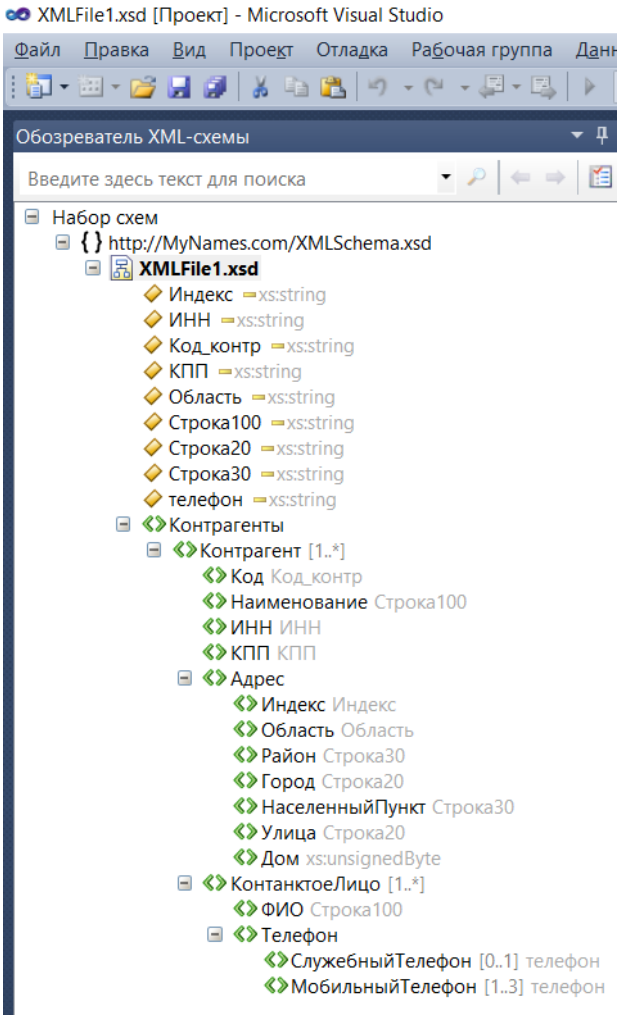


Рис. 6. Схема с созданными глобальными простыми типами данных

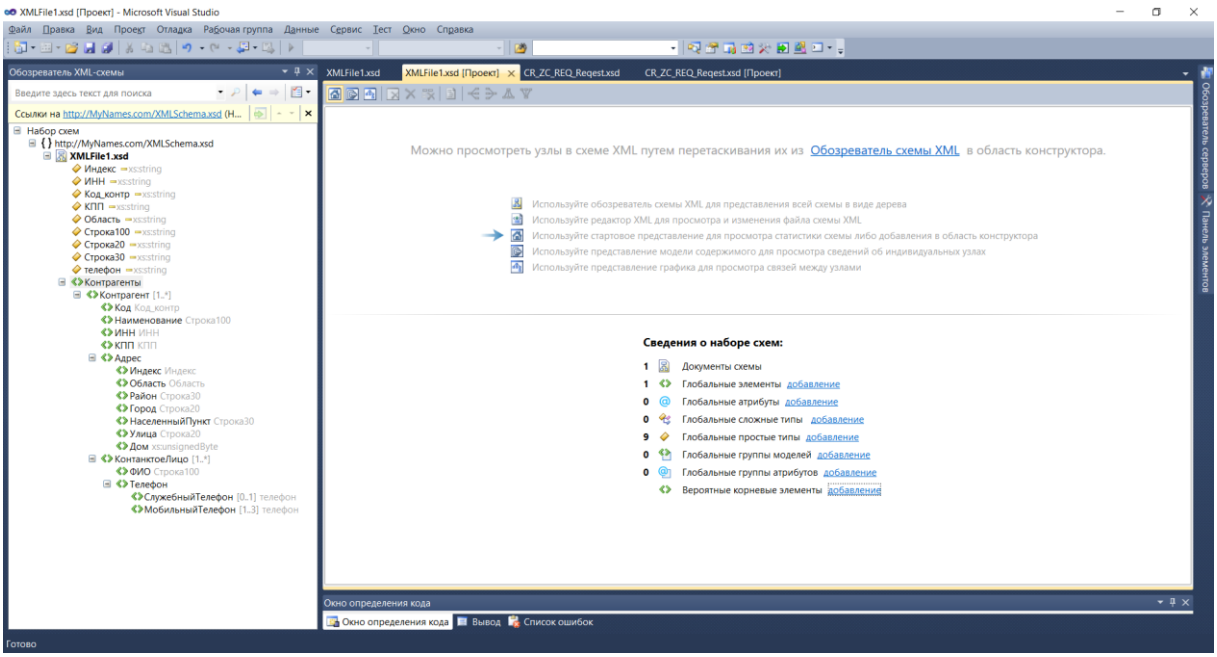


Рис. 7. Стартовое представление схемы

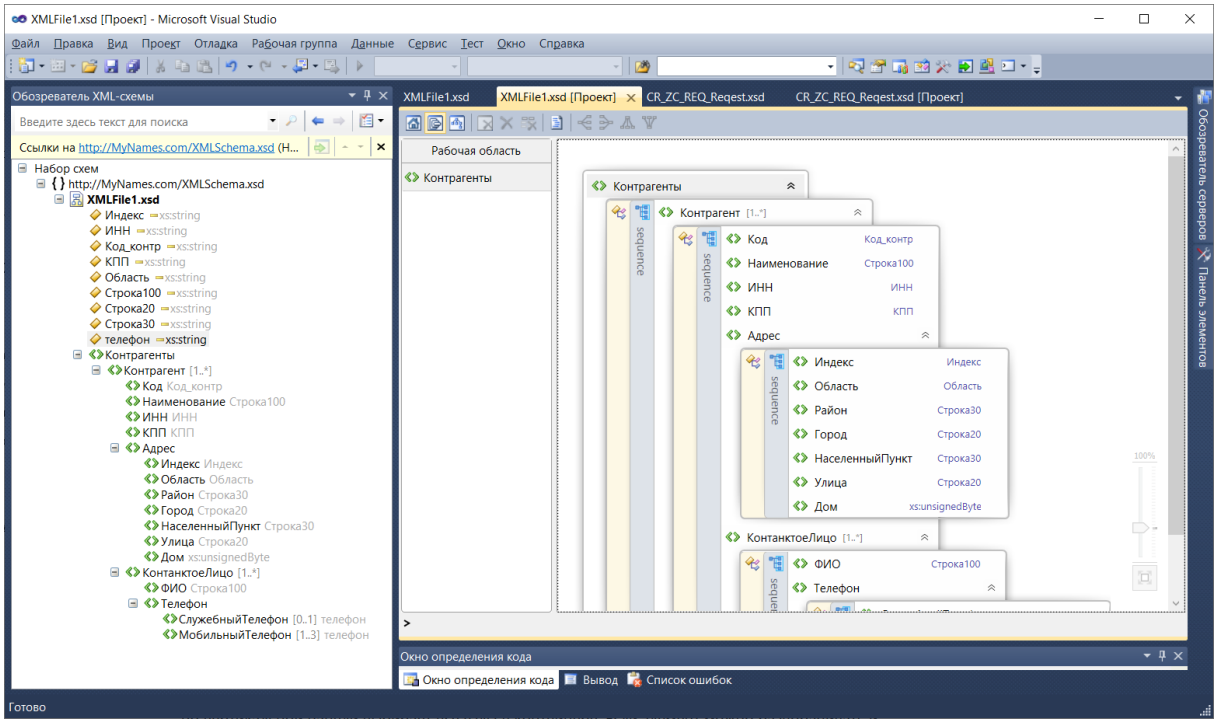


Рис. 8. Модель содержимого схемы

# 5 СОЗДАНИЕ ГРУПП И СЛОЖНЫХ ПОЛЬЗОВАТЕЛЬСКИХ ТИПОВ ДАННЫХ

**Цель работы:** разработать сложные пользовательские типы данных и группы, предназначенные для повторного использования в рамках XSD-схемы.

**Порядок выполнения работы**:

Поскольку группа элементов схемы для описания адреса является устойчивой и часто используемой в различных контекстах (для описания адреса контрагента, сотрудника, адреса организации и пр.), эта группа может быть повторно использована в других схемах. Выделим эту группу в отдельную поименованную группу элементов для этого внесем следующие изменения в код XSD-схемы:

1. Вставьте описание группы в код (в начале или в конце схемы). Данная группа будет определена глобально на уровне всей схемы. Это значит, что ссылка на группу может быть многократно использована в данной схеме. Добавьте аннотации.

```
<xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
```

2. Вставьте ссылку на созданную поименованную группу вместо элемента «Адрес».

```
<xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="Строка100" />
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
              <xs:group ref="Адрес"/>
              <xs:element maxOccurs="unbounded" name="КонтанктоеЛицо">
```

Итоговый код будет выглядеть следующим образом:

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
           targetNamespace="http://MyNames.com/XMLSchema.xsd"
xml:lang="ru">
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="Строка100" />
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
              <xs:group ref="Адрес"/>
              <xs:element maxOccurs="unbounded" name="КонтанктоеЛицо">
                <xs:complexType>
                  <xs:sequence>
                    <xs:element name="ФИО" type="Строка100" />
                    <xs:element name="Телефон">
                      <xs:complexType>
                        <xs:sequence>
```

```
                        <xs:element name="СлужебныйТелефон"
type="телефон"  minOccurs="0" maxOccurs="1"/>
                        <xs:element name="МобильныйТелефон"
type="телефон"  minOccurs="1" maxOccurs="3"/>
                      </xs:sequence>
                    </xs:complexType>
                  </xs:element>
                </xs:sequence>
              </xs:complexType>
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
 </xs:element>
 <xs:annotation>
  <xs:documentation>Блок описания простых типов</xs:documentation>
 </xs:annotation>
  <xs:simpleType name="Код_контр">
   <xs:annotation>
    <xs:documentation>Простой тип для кода контрагент</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
    <xs:length fixed="true" value="4"/>
    <xs:pattern value="[Ю|Ф][0-9]{3}"/>
   </xs:restriction>
 </xs:simpleType>
  <xs:simpleType name="ИНН">
   <xs:annotation>
    <xs:documentation>Простой тип для ИНН</xs:documentation>
   </xs:annotation>
   <xs:restriction base="xs:string">
    <xs:length fixed="true" value="10"/>
    <xs:pattern value="\d{10}"/>
   </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="КПП">
   <xs:annotation>
    <xs:documentation>Простой тип для КПП</xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="9"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="телефон">
    <xs:annotation>
      <xs:documentation>Простой тип для номера телефона</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="(\+7|8)[\s(]*\d{3}[)\s]*\d{3}[\s-]?\d{2}[\s-
]?\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Индекс">
    <xs:annotation>
      <xs:documentation>Простой тип для индекса в блоке
адреса</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="6"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Область">
    <xs:annotation>
      <xs:documentation>Справочник областей</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Московская"/>
      <xs:enumeration value="Ленинградская"/>
      <xs:enumeration value="Владимирская"/>
      <xs:enumeration value="Тверская"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка30">
    <xs:annotation>
      <xs:documentation>Строка не более 30 любых
символов</xs:documentation>
```

```
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка20">
    <xs:annotation>
      <xs:documentation>Строка не более 20 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="20"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка100">
    <xs:annotation>
      <xs:documentation>Строка не более 100 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
</xs:schema>
```

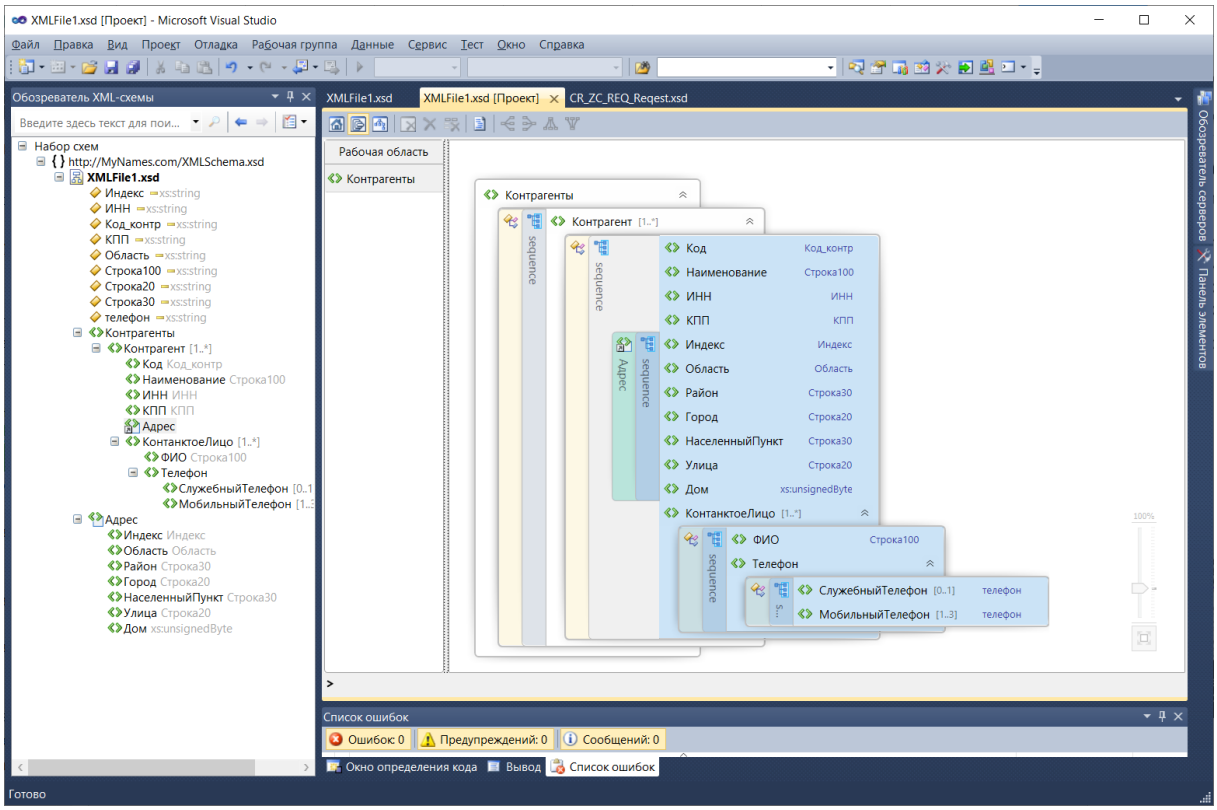


Рис. 9. Модель содержимого обновленной схемы

Создадим глобально объявленный сложный тип данных для описания реквизитов контактного лица. Для этого:

3. Вставьте описание поименованного сложного типа в начале или в конце схемы. Добавьте аннотации.

```
<xs:complexType name="КонтактноеЛицо">
    <xs:annotation>
      <xs:documentation>Сложный тип для описания контактного
лица</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ФИО" type="Строка100" />
      <xs:element name="Телефон">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="СлужебныйТелефон" type="телефон"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="МобильныйТелефон" type="телефон"
minOccurs="1" maxOccurs="3"/>
          </xs:sequence>
        </xs:complexType>
```

```
      </xs:element>
    </xs:sequence>
  </xs:complexType>
```

4. Добавьте ссылку на созданный сложный тип данных в схеме

```
<xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="Строка100" />
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
              <xs:group ref="Адрес"/>
              <xs:element  name="КонтанктоеЛицо" type="КонтактноеЛицо"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

Итоговый код схемы представлен ниже:

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
targetNamespace="http://MyNames.com/XMLSchema.xsd" xml:lang="ru">
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Контрагент" maxOccurs="unbounded">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="Строка100" />
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
              <xs:group ref="Адрес"/>
              <xs:element  name="КонтанктоеЛицо" type="КонтактноеЛицо"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
```

```
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:annotation>
    <xs:documentation>Блок описания простых типов</xs:documentation>
  </xs:annotation>
   <xs:simpleType name="Код_контр">
    <xs:annotation>
      <xs:documentation>Простой тип для кода контрагент</xs:documentation>
     </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="4"/>
      <xs:pattern value="[Ю|Ф][0-9]{3}"/>
    </xs:restriction>
  </xs:simpleType>
   <xs:simpleType name="ИНН">
    <xs:annotation>
      <xs:documentation>Простой тип для ИНН</xs:documentation>
     </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="10"/>
      <xs:pattern value="\d{10}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="КПП">
    <xs:annotation>
      <xs:documentation>Простой тип для КПП</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="9"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="телефон">
    <xs:annotation>
      <xs:documentation>Простой тип для номера телефона</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="(\+7|8)[\s(]*\d{3}[)\s]*\d{3}[\s-]?\d{2}[\s-
]?\d{2}"/>
```

```
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Индекс">
    <xs:annotation>
      <xs:documentation>Простой тип для индекса в блоке
адреса</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:length fixed="true" value="6"/>
      <xs:pattern value="\d{9}"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Область">
    <xs:annotation>
      <xs:documentation>Справочник областей</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Московская"/>
      <xs:enumeration value="Ленинградская"/>
      <xs:enumeration value="Владимирская"/>
      <xs:enumeration value="Тверская"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка30">
    <xs:annotation>
      <xs:documentation>Строка не более 30 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="30"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="Строка20">
    <xs:annotation>
      <xs:documentation>Строка не более 20 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="20"/>
    </xs:restriction>
```

```
  </xs:simpleType>
  <xs:simpleType name="Строка100">
    <xs:annotation>
      <xs:documentation>Строка не более 100 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
    <xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
  <xs:complexType name="КонтактноеЛицо">
    <xs:annotation>
      <xs:documentation>Сложный тип для описания контактного
лица</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ФИО" type="Строка100" />
      <xs:element name="Телефон">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="СлужебныйТелефон" type="телефон"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="МобильныйТелефон" type="телефон"
minOccurs="1" maxOccurs="3"/>
          </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

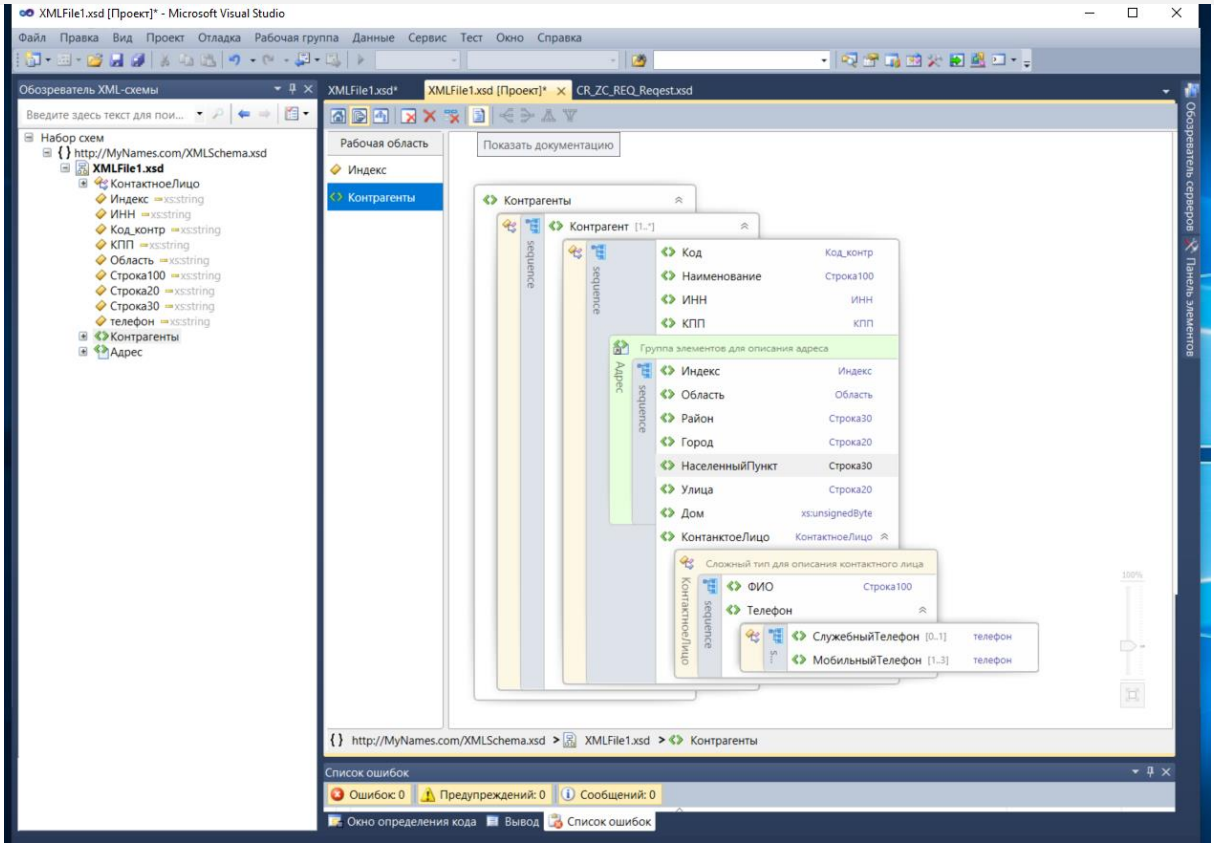


Рис. 10. Модель содержимого обновленной схемы (включена опция «Показать документацию»)

# 6 СОЗДАНИЕ МОДУЛЬНОЙ СХЕМЫ (ЧАСТЬ 1)

**Цель работы:** освоить технологию создания модульных XSD-схем с использованием операции include.

**Порядок выполнения работы**:

Создадим две копии схемы, полученной в п.5 с именами XMLFile1Import.xsd и XMLFile1Include.xsd. для каждой схемы создайте собственную папку, в которой будет храниться модульная схема и все входящие в нее подсхемы).

Выделим в отдельные подсхемы часто используемые простые типы данных (например, тип «телефон» и «Строка 100»). Ссылки на эти типы данных могут использоваться многократно в различных транзакциях при передачи данных между корпоративными приложениями, в том числе и при передаче данных о контрагентах. Основная схема (мастер схема) будет ссылаться на внешние подсхемы, при этом при использовании include включаемая подсхема становится частью пространства имен основной схемы. Данный вариант используется, если основная схема имеет единый контекст (единое пространство имен).

Для создания отдельной схемы для простого типа «телефон» выполним следующие действия:

1. Создадим новую схему в MS Visual Studio (команда Файл / Создать / Файл, в форме «Создать файл» выбрать XML-схема). В стартовом представлении перейдем редактор XML.
2. Сохраним схему в рабочем каталоге под именем TelephonType.xsd (команда Файл / Сохранить…как).
3. Откроем файл XMLFile1Include.xsd и **вырежем** код с описанием простого типа «телефон» из исходной схемы (см. раздел 5) и вставим его в TelephonType.xsd. Отредактируем атрибуты корневого тега схемы. Получим следующий код (см. ниже). Сохраним изменения.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xml:lang="ru">
  <xs:simpleType name="телефон">
    <xs:annotation>
      <xs:documentation>Простой тип для номера телефона</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="(\+7|8)[\s(]*\d{3}[)\s]*\d{3}[\s-]?\d{2}[\s-
]?\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
```

```
</xs:schema>
```

4. Вставим в мастер схему ссылку на подсхему.

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema attributeFormDefault="unqualified"
elementFormDefault="unqualified"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
           targetNamespace="http://MyNames.com/XMLSchema.xsd"
xml:lang="ru">
  <xs:include schemaLocation="TelephonType.xsd"/>
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
```

В обозревателе XML-схемы видно, что включенная подсхема становится частью пространства имен мастер схемы (Рис. 11).


Рис. 11. Модель содержимого мастер схемы

Можно создать несколько уровней вложенности подсхем. Для разработки примера схемы с несколькими уровнями ссылок:

1. Создадим подсхему ContactType.xsd и перенесем в нее сложный тип «КонтактноеЛицо» из схемы XMLFile1Include.xsd.
2. Поскольку сложный тип «КонтактноеЛицо» ссылается на простой тип «телефон», импортирует подсхему TelephonType.xsd в ContactType.xsd. Поскольку подсхема ContactType.xsd ссылается на простой тип данных Строка 100, описание этого типа данных перенесено из мастер схемы в данную подсхему. Получим следующий код для схемы ContactType.xsd.

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

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xml:lang="ru">
  <xs:include schemaLocation="TelephonType.xsd"/>
  <xs:complexType name="КонтактноеЛицо">
    <xs:annotation>
      <xs:documentation>Сложный тип для описания контактного
лица</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ФИО" type="Строка100" />
      <xs:element name="Телефон">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="СлужебныйТелефон" type="телефон"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="МобильныйТелефон" type="телефон"
minOccurs="1" maxOccurs="3"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  <xs:simpleType name="Строка100">
    <xs:annotation>
      <xs:documentation>Строка не более 100 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

3. В мастер схеме вставим ссылку на подсхему ContactType.xsd, исключим из мастер схемы описание простого типа «Строка100». Описание этого типа будет унаследовано из включенной подсхемы. Структура мастер схемы, содержащей подсхемы нескольких уровней представлена ниже.

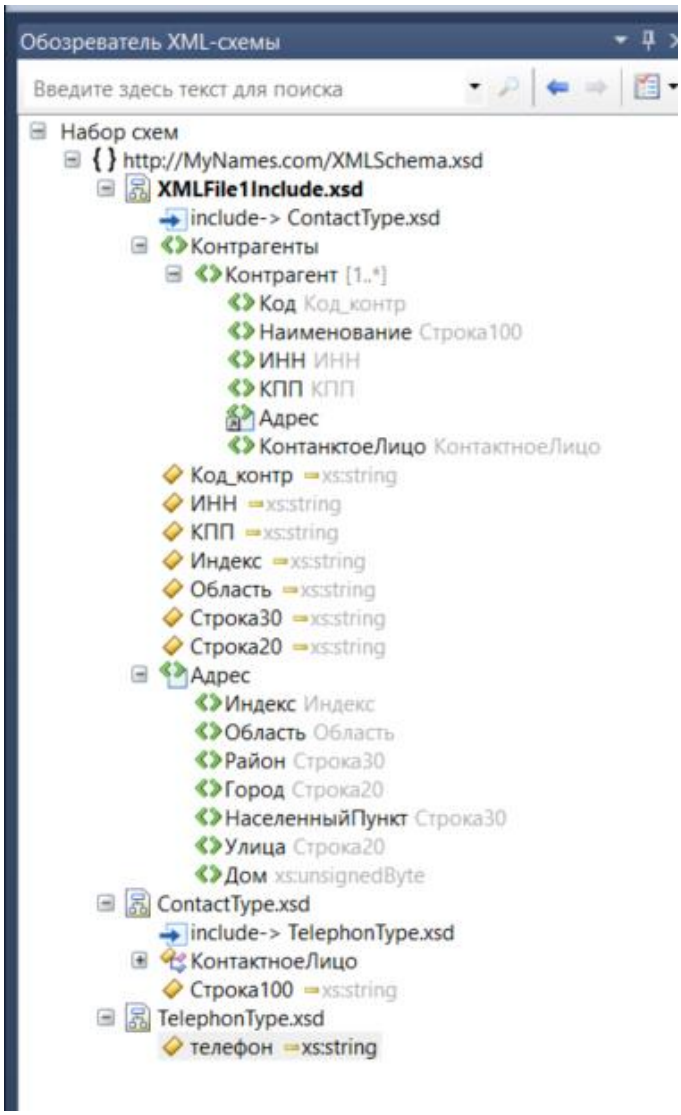


Рис. 12. Модель содержимого мастер схемы

4. Самостоятельно вынесите в отдельные подсхемы описания простых типов данных и блока адреса.

Подсхема для блока адреса будет иметь вид, представленный ниже. Подсхама включает ссылки на простые типы данных для описания индекса, области др. используемых простых типов.

```
?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
           targetNamespace="http://MyNames.com/XMLSchema.xsd"
xml:lang="ru">
  <xs:include schemaLocation="IndexType.xsd"/>
  <xs:include schemaLocation="RegionType.xsd"/>
```

```
  <xs:include schemaLocation="String30Type.xsd"/>
  <xs:include schemaLocation="String20Type.xsd"/>
  <xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
</xs:schema>
```

Структура мастер схемы после всех преобразований показана на Рис. 13.

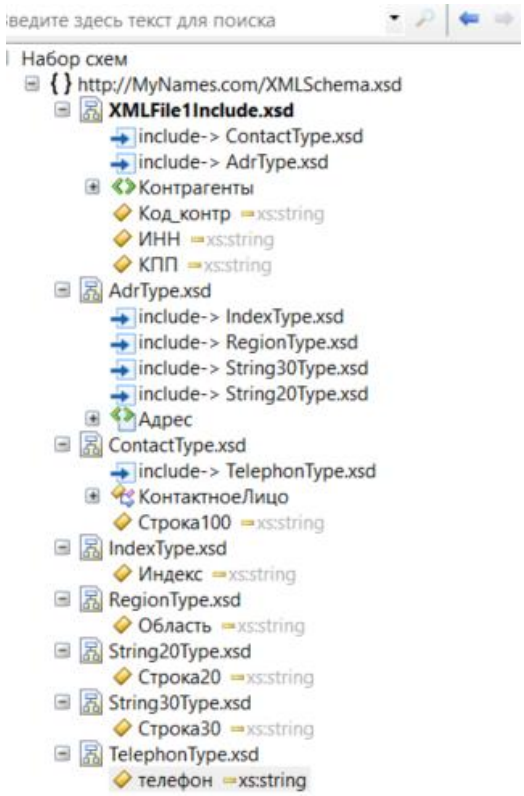


Рис. 13. Итоговая модель содержимого мастер схемы

Обратите внимание на то, что все элементы мастер схемы и включенных подсхем принадлежат одному пространству имен (пространству имен главной схемы). Таким образом, в данном разделе показан пример разработки модульной XSD-схемы для интеграционных проектов с пространством имен типа "хамелеон" - главной XML-схеме присваивается targetNamespace, а вспомогательным XML-схемам не присваивается никакого targetNamespace (XML-схемы без targetNamespace используют targetNamespace главной XML-схемы при объединении подобно хамелеону). «Гомогенное пространство имен» можно получить, если всем XML-схемам присваивается одинаковый targetNamespace.

# 7 СОЗДАНИЕ МОДУЛЬНОЙ СХЕМЫ (ЧАСТЬ 2)

**Цель работы:** освоить технологию создания модульных XSD-схем с использованием операции import.

**Порядок выполнения работы**:

1. Откройте файл схемы XMLFile1Import.
2. Создайте подсхемы для описания отдельных типов данных и групп элементов аналогично разделу 6. В каждой подсхеме укажите собственное пространство имен с уникальным URI и префиксом:
   - Создадим подсхему для описания простого типа номера телефона. Определим для этой подсхемы собственное пространство имен http://Telephon.com и укажем это пространство имен как пространство имен по умолчанию.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" xml:lang="ru"
xmlns="http://Telephon.com" elementFormDefault="qualified"
targetNamespace="http://Telephon.com">
  <xs:simpleType name="телефон">
    <xs:annotation>
```

```
      <xs:documentation>Простой тип для номера телефона</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:pattern value="(\+7|8)[\s(]*\d{3}[)\s]*\d{3}[\s-]?\d{2}[\s-
]?\d{2}"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

- Создадим подсхему для описания простого типа String100. Определим для этой подсхемы собственное пространство имен http://String100.com и укажем это пространство имен как пространство имен по умолчанию.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://String100.com" targetNamespace="http://String100.com"
elementFormDefault="qualified">
  <xs:simpleType name="Строка100">
    <xs:annotation>
      <xs:documentation>Строка не более 100 любых
символов</xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:string">
      <xs:maxLength value="100"/>
    </xs:restriction>
  </xs:simpleType>
</xs:schema>
```

- Создаем подсхему для описания сложного типа «КонтактноеЛицо» (в файле ContactType.xsd). Определим для этой подсхемы собственное пространство имен http://Contact.com и укажем это пространство имен как пространство имен по умолчанию. Определим в корневом теге схемы пространства имен для подсхем с указанием уникального префикса для каждой подсхемы. Выполним импорт подсхем, указав для каждой подсхемы расположение файла подсхемы (в атрибуте

schemaLocation) и пространство имен подсхемы (в атрибуте namespace). Ссылки на простые типы, определенные в подсхемах используют заданные префиксы. При вводе ссылочных типов поддерживается технология IntelliSense (см. выше).

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://Contact.com" targetNamespace="http://Contact.com"
           xmlns:te="http://Telephon.com" xmlns:s100="http://String100.com"
           elementFormDefault="qualified" xml:lang="ru" >
  <xs:import schemaLocation="TelephonType.xsd"
namespace="http://Telephon.com"/>
  <xs:import schemaLocation="String100.xsd"
namespace="http://String100.com"/>
   <xs:complexType name="КонтактноеЛицо">
    <xs:annotation>
      <xs:documentation>Сложный тип для описания контактного
лица</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="ФИО" type="s100:Строка100" />
      <xs:element name="Телефон">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="СлужебныйТелефон" type="te:телефон"
minOccurs="0" maxOccurs="1"/>
            <xs:element name="МобильныйТелефон" type="te:телефон"
minOccurs="1" maxOccurs="3"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
  </xs:schema>
```

- Собираем мастер-схему (файл XMLFile1Import). Мастер схема должна содержать ссылки на пространства имен всех включенных в нее подсхем с указанием уникальных префиксов (в данном примере это два пространства имен http://Contact.com и

http://String100.com, ссылку на пространство имен http://Telephon.com вставлять не надо, т.к. в мастер схеме тип, описанный в подсхеме TelephonType.xsd, не используется). Выполним импорт подсхем, указав для каждой подсхемы расположение файла подсхемы (в атрибуте schemaLocation) и пространство имен подсхемы (в атрибуте namespace). Расставляем ссылки на простые и сложные типы, определенные в подсхемах (ссылки используют префиксы). Если в списке доступных для выбора типов не отображается тип, описанный в подсхеме с префиксом, значит импорт подсхем выполнен с ошибкой.

*Примечание*

*Для отображения списка ошибок в MS Visual Studio необходимо выполнить команду Вид / Список ошибок.*

Фрагмент кода мастер схемы, использующей импортированные подсхемы, представлен ниже.

```
<?xml version="1.0" encoding="windows-1251"?>
<xs:schema elementFormDefault="qualified"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
 xmlns:con="http://Contact.com" xmlns:s100="http://String100.com"
 targetNamespace="http://MyNames.com/XMLSchema.xsd" xml:lang="ru">
  <xs:import schemaLocation="ContactType.xsd"
namespace="http://Contact.com"/>
  <xs:import schemaLocation="String100.xsd"
namespace="http://String100.com"/>
  <xs:element name="Контрагенты">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="unbounded" name="Контрагент">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="Код" type="Код_контр"/>
              <xs:element name="Наименование" type="s100:Строка100"/>
              <xs:element name="ИНН" type="ИНН" />
              <xs:element name="КПП" type="КПП"/>
```

```
                <xs:group ref="Адрес"/>
                <xs:element  name="КонтактноеЛицо"
type="con:КонтактноеЛицо"/>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
```

Модель содержимого данного варианта мастер схемы представлена на Рис. 14. Элементы всех импортированных подсхем сохранили свои пространства имен после импорта их в мастер-схему.

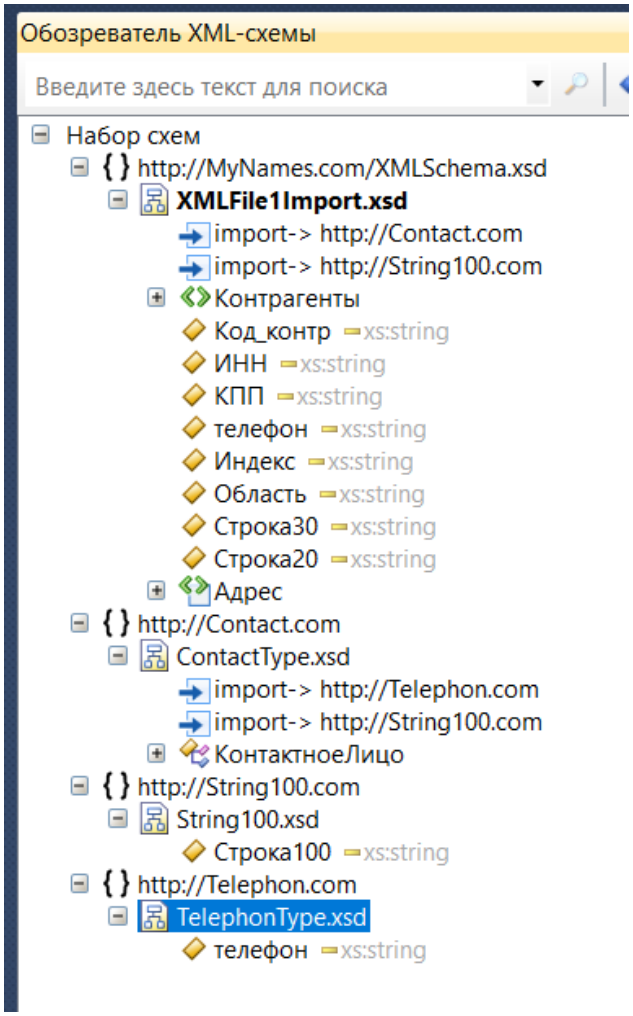


Рис. 14. Модель содержимого мастер схемы

Действуя аналогично, самостоятельно вынесите в отдельные подсхемы описания простых типов и блока адреса. Выполните импорт созданных подсхем в мастер схему XMLFile1Import.

С помощью импорта подсхем формируется «гетерогенное пространство имен». Такой подход используется, если различные подсхемы контролируются различными приложениями и необходимо понимать происхождение тех-или иных данных после их объединения.

# 8 СОЗДАНИЕ МОДУЛЬНОЙ СХЕМЫ (ЧАСТЬ 3)

**Цель работы:** освоить технологию создания модульных XSD-схем с использованием операции redefine.

*Примечание*

*Redefine используется в XML-схемах для получения доступа к компонентам в других XML-схемах и одновременно дает возможность внести изменения в определения импортируемых компонентов. Таким образом, элемент <redefine> выполняет двойную функцию:*

- *он выполняет неявный <include>, что позволяет иметь доступ ко всем компонентам во вспомогательных схемах;*
- *он дает возможность внести какое-то число (ноль или более) изменений в определения импортируемых компонентов, то есть расширить определения компонентов или наоборот наложить дополнительные ограничения на определения компонентов.*

Пусть необходимо переопределить ранее созданный простой тип «область», добавив ограничения на длину строки.

**Порядок выполнения работы**:

1. Сделайте копию папки с XSD-схемами, полученными в разделе 2.
2. Откройте схему, использующую простой тип «область», это схема AdrType.xsd включающая несколько подсхем с помощью include.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
           targetNamespace="http://MyNames.com/XMLSchema.xsd"
xml:lang="ru">
  <xs:include schemaLocation="IndexType.xsd"/>
  <xs:include schemaLocation="RegionType.xsd"/>
  <xs:include schemaLocation="String30Type.xsd"/>
  <xs:include schemaLocation="String20Type.xsd"/>
  <xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
    <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
</xs:schema>
```

3. Используем redefine для подсхемы RegionType.xsd. Включим в этот тег дополнительное ограничение на длину строки.

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://MyNames.com/XMLSchema.xsd"
           targetNamespace="http://MyNames.com/XMLSchema.xsd"
xml:lang="ru">
  <xs:include schemaLocation="IndexType.xsd"/>
  <xs:include schemaLocation="String30Type.xsd"/>
  <xs:include schemaLocation="String20Type.xsd"/>
  <xs:redefine schemaLocation="RegionType.xsd">
    <xs:simpleType name="Область">
      <xs:restriction base="Область">
```

```
        <xs:length value="25"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:redefine>
       <xs:group name="Адрес">
    <xs:annotation>
      <xs:documentation>Группа элементов для описания
адреса</xs:documentation>
    </xs:annotation>
        <xs:sequence>
      <xs:element name="Индекс" type="Индекс"/>
      <xs:element name="Область" type="Область"/>
      <xs:element name="Район"  type="Строка30"/>
      <xs:element name="Город" type="Строка20"/>
      <xs:element name="НаселенныйПункт" type="Строка30" />
      <xs:element name="Улица" type="Строка20"/>
      <xs:element name="Дом" type="xs:unsignedByte" />
    </xs:sequence>
  </xs:group>
</xs:schema>
```

Переопределение является очень мощным инструментом для разработки модульных XSD-схем, позволяющим изменить \дополнить конструкции XSD.

Модель содержимого мастер-схемы, использующей redefine представлена на Рис. 15. Все элементы после переопределения относятся к единому пространству имен.

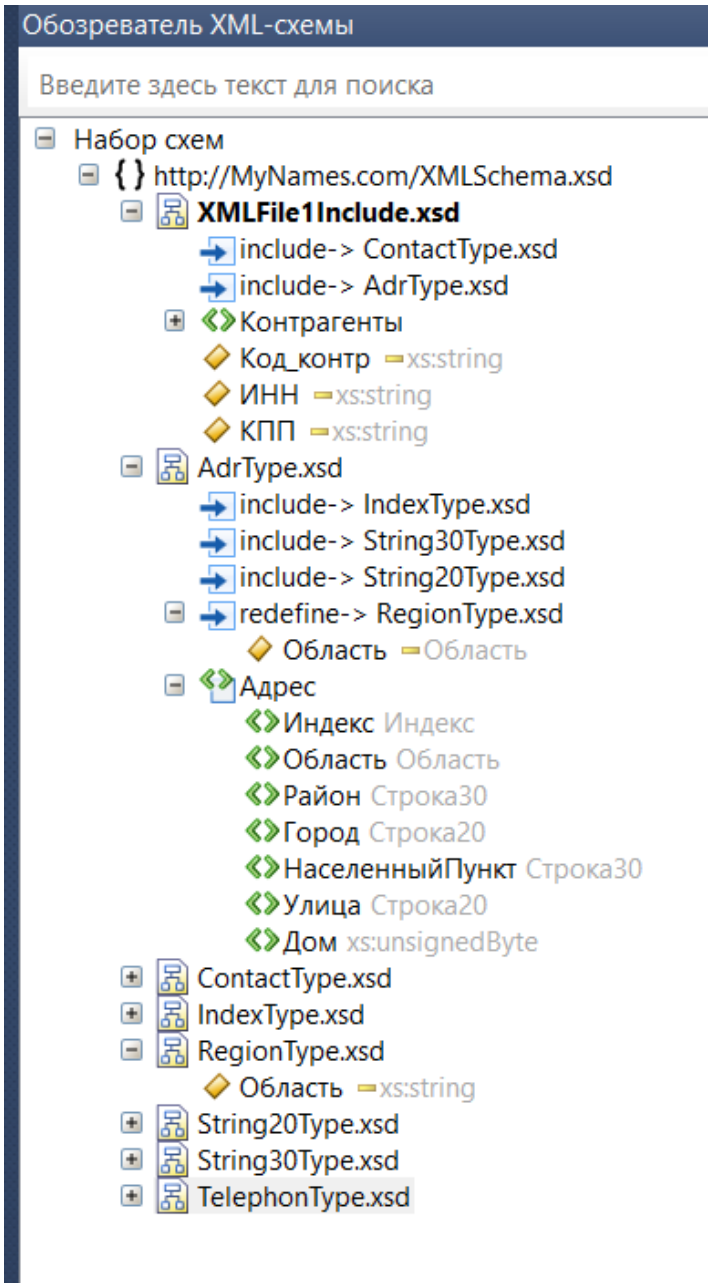


Рис. 15. Модель содержимого мастер схемы (redefine)

## СПИСОК ИСТОЧНИКОВ


1. ГОСТ Р ИСО/МЭК 7498-1-99 Информационная технология. Взаимосвязь открытых систем. Базовая эталонная модель. Часть 1. Базовая модель
2. ГОСТ Р 54878-2011 Электронный обмен данными в управлении, торговле и на транспорте (EDIFACT). Принципы формирования файлов XML схемы (XSD) на основе инструкций по реализации EDI (FACT)
3. Морозова, О.А. Интеграция корпоративных информационных систем = Enterprise information systems integration. Manual: Учебное пособие / О.А. Морозова; Финуниверситет, Каф. бизнес-информатики. — М.: Финуниверситет, 2014. — 140 с. — Имеется электронная версия: Электронные текстовые данные ( 1 файл: 1 Мб). — Свободный доступ из сети Интернет (чтение, печать, копирование). — <URL:http://elib.fa.ru/fbook/Morozova_integr.pdf>.
4. Хоп Г., Вульф Б. Шаблоны интеграции корпоративных приложений.: Пер с англ. - М.: ООО «И.Д. Вильямс», 2007.-672 с.
5. Бин Д. XML для проектировщиков. Повторное использование и интеграция. – М.: КУДИНЦ-ОБРАЗ, 2004.-256 с.
6. Машнин Т.С. Web-сервисы Java.- СПб.:БХВ-Петербург, 2012.- 560 с.
7. Берсон А. Дубов Л. Управление мастер-данными. – Изд. Дом «Ноосфера», 2017, - 365 с.
8. Extensible Markup Language (XML) 1.0 (Fifth Edition) https://www.w3.org/TR/xml/
9. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures https://www.w3.org/TR/xmlschema11-1/
10. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes https://www.w3.org/TR/xmlschema11-2/
11. XSL Transformations (XSLT) Version 2.0 (Second Edition) https://www.w3.org/TR/2009/PER-xslt20-20090421/
12. World Wide Web Consortium. Web Services Discription Language (WSDL) 1.1. http://www.w3.org/TR/wsdl
13. World Wide Web Consortium. SOAP Current status. http://www.w3.org/standards/techs/soap#w3c_all
14. Business Process Execution Language for Web Services Version 1.1. http://xml.coverpages.org/BPELv11-May052003Final.pdf
15. Школы консорциума W3C / XML http://xml.nsu.ru/xml/xml_home.xml
16. Open Applications Group Integration Specification (OAGIS) https://www.service-architecture.com/articles/xml/oagis.html
17. ASC X12 Standard Interpretations http://www.x12.org/rfis/
18. Enterprise Connectivity Patterns: Implementing integration solutions with IBM's Enterprise Service Bus products. http://www.ibm.com/developerworks/webservices/library/ws-

enterpriseconnectivitypatterns/index.html?S_TACT=105AGX99&S_CMP=CP

19. Практика использования пространств имен XML в проектах, содержащих несколько XML-схем. http://www.interface.ru/home.asp?artId=21058

## ПРИЛОЖЕНИЕ 1

Варианты заданий

| Вариант | Содержание документа |
|---|---|
| 1. | Данные о клиентах |
| 2. | Данные о поставщиках |
| 3. | Данные о продуктах |
| 4. | Данные о сотрудниках |
| 5. | Сведения об услугах |
| 6. | Сведения о товарах |
| 7. | Сведения о проекте |
| 8. | Данные накладной |
| 9. | Данные заказа |
| 10. | Данные договора |
| 11. | Данные счета клиента |
| 12. | Данные о платеже |
| 13. | Данные медицинской карты |
| 14. | Сведения о налогоплательщике |
| 15. | Данные о предоставленной услуге |
| 16. | Данные об успеваемости |
| 17. | Данные об оплате коммунальных услуг |
| 18. | Данные управленческого отчета |
| 19. | Данные тайм-шита |
| 20. | Данные оборотно-сальдовой ведомости |
| 21. | Данные об отгрузке товаров |
| 22. | Данные о репертуаре |
| 23. | Данные об акционерах |
| 24. | Данные о ценных бумагах |
| 25. | Данные библиотечного каталога |
| 26. | Данные о комплектующих |

## ПРИЛОЖЕНИЕ 2

Требования к корректности XML-документов

Синтаксически правильный XML-документ:

1. Начинается со строки объявления, которая ограничивается тегами <?xml и ?>.
2. Представляется в виде дерева элементов, каждый из которых может иметь набор атрибутов, а также содержать другие элементы или текст.
3. Должен содержать один корневой элемент.
4. Элемент представляется в документе XML с помощью открывающего тэга (<>) и закрывающего тэга (</>).Открывающий тэг записывается в формате <ИмяЭлемента>, а закрывающий тэг в формате </ИмяЭлемента>.
5. Элементы могут быть вложены друг в друга, но не могут пересекаться.
6. Имя элемента не может содержать пробелов.
7. Содержимым элемента могут быть символьные данные (текст), другие элементы (также известные как дочерние элементы), а также оно может отсутствовать (пустой элемент).
8. Для пустого элемента допустима запись <ИмяЭлемента/>
9. Элемент может содержать любое число атрибутов, содержащих дополнительную информацию о данных, которые представляет элемент.
10. Атрибуты указываются в виде пар название-значение в открывающем тэге элемента.
11. Значения атрибутов заключаются в кавычки.
12. Названия атрибутов уникальны в рамках одного элемента (в одном элементе не может быть двух атрибутов с одинаковым именем).

## ПРИЛОЖЕНИЕ 3

Язык определения XML-схем" (XML Schema Definition Language). Версия 1.1 (краткое руководство)

Корневым элементом всегда является элемент <schema>. Описание атрибутов элемента <schema> приводится в таблице.

| Атрибут | Обязательный | Описание |
|---|---|---|
| elementFormDefault | Нет | Принимает значения "qualified" и "unqualified" (по умолчанию)<br>Значение "qualified" указывет на то, что элементы документа должны уточняться префиксом пространства имен. |
| attributeFormDefault | Нет | Принимает значения "qualified" и "unqualified" (по умолчанию)<br>Значение "qualified" указывет на то, что атрибуты элементов документа должны уточняться префиксом пространства имен. |
| xmlns:xs | Да | Всегда принимает значение xmlns:xs="http://www.w3.org/2001/XMLSchema Указывает на пространство имен языка XMLSchema для элементов и типов данных схемы. При наличии префикса (в данном примере xs)все элементы и типы данных схемы должны уточняться этим префиксом (xs: schema). Если префикс отсутствует атрибут xmlns указывает для схемы пространство имен по умолчанию.<br>Элемент <schema> может содержать несколько атрибутов xmlns с разными префиксами, указывающими на используемые в документе пространства имен. |
| targetNamespace | Нет | Пространство имен для элементов XML-документа, определенных данной схемой. |
| version | Нет | Версия схемы |
| xml:lang | Нет | Язык для всех комментариев схемы |

Коневой элемент <schema> может содержать следующие дочерние элементы:

- <element> - используется для определения элементов XML-документа,
- <attribute> - используется для определения атрибутов XML-документа,

- <group> - необходим для определения группы элементов, предназначенной для повторного использования в рамках схемы по ссылке на имя группы,
- <attributeGroup> - используется для определения атрибутов группы элементов.
- <annotation> - позволяет включать в XML-документ документацию,
- <import> - позволяет использовать компоненты указанной внешней схемы в основнойц схеме (обеспечивает модульность схем),
- <include> - добавляет все компоненты указанной внешней схемы в основную схему (обеспечивает модульность схем),
- < notation> - содержит определение нотации, описывающей формат не-XML данных в XML-документе,
- <redefine> - переопределяет компоненты внешней схемы, имеющей такое же пространство имен, что и основная схема,
- <simpleType> - объявляет простой тип содержимого элемента. Элементы с простым типом данных могут содержать только символьные данные и не могут включать атрибуты и дочерние элементы.
- <complexType> - объявляет сложный тип содержимого элемента, которые могут включать атрибуты и другие элементы.

XML Schema поддерживает три основные категории типов данных:

1. Предопределенные примитивные типы (это фундаментальные типы данных, на которые можно ссылаться и применять их к элементам и атрибутам). Примерами примитивных типов данных являются String, Float, Double, Time, Date, Decimal, AnyURI.
2. Предопределенные производные типы (это встроенные типы, полученные на основании примитивных типов). Примерами

производных типов данных являются Integer, Long, Byte, Short, nonPositiveInteger, nonNegativeInteger, ID и т.д.

3. Нестандартные типы (это определяемые пользователем типы данных, которые создаются на основании примитивных или производных типов путем введения дополнительных ограничений). Поддержка нестандартных типов данных исключительно полезна для верификации данных с учетом бизнес-логики.

Для описания элементов и атрибутов, имеющих предопределенные (примитивные и производные) типы данных[2] в XML Schema используются следующие синтаксические конструкции:

```
    <xs:element name="ИмяЭлемента" type="ТипДанных"
/>
    <xs:attribute name="ИмяАтрибута"
type="ТипДанных"/>
```

Дополнительно для элементов и атрибутов можно указать атрибуты fixed или default для задания фиксированных значений элементов\атрибутов или значений, заданных по умолчанию.

```
    <xs:element name="Пример" type="xs:string"
default="Пример описания элемента"/>
```

Если необходимо описать нестандартный тип данных для элемента или атрибута, это следует делать с помощью тега <SympleType>, описав в нем новый тип данных.

```
    <xs:element name="ИмяЭлемента">
        <xs:simpleType>
            Описание нестандартного типа данных

        </xs:simpleType>
    </xs:element>
```

Новые нестандартные простые типы данных получают путем:

[2] А также заранее определенные нестандартные типы данных (см. ниже)

- сужения (restriction) встроенного или ранее определенного простого типа с помощью задания дополнительных ограничений,
- объединения (union) простых типов,
- использования списка (list) простых типов,

Пример использования нового простого типа данных, полученного путем сужения предопределенного типа (на базовый тип String накладываются ограничения на максимально и минимально допустимую длину строки).

```
    <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="20"/>
                    <xs:minLength value="10"/>
                </xs:restriction>
    </xs:simpleType>
```

Пример использования нового простого типа данных, полученного путем объединения базовых типов (элемент или атрибут смогут принимать неотрицательные или неположительные целые значения).

```
    <xs:simpleType>
    <xs:union memberTypes="xs:nonNegativeInteger
xs:nonPositiveInteger" />
    </xs:simpleType>
```

Пример использования списка простых типов (атрибут shoeSizes объявляется в качестве списка, содержащего десятичные значения 10.5, 9, 8 и 11).

```
    <xs:simpleType name="Sizes">
          <xs:restriction base="xs:decimal">
             <xs:enumeration value="10.5"/>
             <xs:enumeration value="9"/>
             <xs:enumeration value="8"/>
       <xs:enumeration value="11"/>
          </xs:restriction>
     </xs:simpleType>

       <xs:attribute name="shoeSizes">
          <xs:simpleType>
```

```
            <xs:list itemType="Sizes"/>
        </xs:simpleType>
    </xs:attribute>
```

Язык XML Schema использует различные типы ограничений на данные (см. таблицу):

- Ограничения длины (количество символов),
- Ограничения значений (наибольшее и наименьшее значение, как диапазон или порог),
- Ограничения количества цифр десятичного числа (общее количество цифр или количество цифр после запятой),
- Список допустимых значений,
- Шаблоны,
- Обработка символов пробела.

Примеры использования различных ограничений представлены в таблице.

| Тип ограничения | Теги, задающие ограничения |
|---|---|
| Ограничения длины | <length> - фиксированное количество символов;<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:length value="4"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br><br><maxLength> - наибольшая длина значений определяемого типа;<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:maxLength value="20"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br><br><minLength> - наименьшая длина значений определяемого типа;<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:minLength value="2"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>` |

| | |
|---|---|
| Границы значений | <maxInclusive> - максимально допустимое значение<br><minInclusive>- минимально допустимое значение<br>В примере допустимы целые значения в диапазоне от 5 до 10, включая 5 и 10.<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:integer">`<br>`<xs:maxInclusive value="10"/>`<br>`<xs:minInclusive value="5"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br><br><maxExclusive> - максимально допустимое значение<br><minExclusive>- минимально допустимое значение<br>В примере допустимы целые значения в диапазоне от 5 до 10, исключая 5 и 10.<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:integer">`<br>`<xs:maxExclusive value="10"/>`<br>`<xs:minExclusive value="5"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>` |
| Ограничения количества и типа цифр | <totalDigits> - общее количество цифр в определяемом числовом типе - сужении типа decimal;<br><fractionDigits> - количество цифр в дробной части числа;<br><br>`<xs:simpleType>`<br>`<xs:restriction base="xs:decimal">`<br>`<xs:totalDigits value="8"/>`<br>`<xs:fractionDigits value="3"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>` |
| Список допустимых значений | <enumeration> - элемент списка допустимых значений;<br><br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:enumeration value="один"/>`<br>`<xs:enumeration value="два"/>`<br>`<xs:enumeration value="три"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>` |
| Шаблоны | <pattern> - регулярное выражение, используется для ограничения внешнего вида или формы значений данных.<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string"`<br>`<xs:pattern value="[Ю\|Ф][0-9]{3}"/>`<br>`</xs:restriction>`<br>`</xs:simpleType` |

| | |
|---|---|
| Обработка символов пробела | <whitespace> - применяется при сужении типа string и определяет способ преобразования пробельных символов (пробел, табуляция, перевод строки)<br>Используется в трех вариантах:<br>1. Все пробельные символы сохраняются<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:whiteSpace value="preserve"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br>2. XML-процессор заменяет все пробельные символя на пробелы.<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:whiteSpace value="replace"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>`<br>3. XML-процессор заменяет пробельные символы пробелами,затем убирает начальные и конечные пробелы, а из нескольких подряд идущих пробелов оставляет только один<br>`<xs:simpleType>`<br>`<xs:restriction base="xs:string">`<br>`<xs:whiteSpace value="collapse"/>`<br>`</xs:restriction>`<br>`</xs:simpleType>` |

Элементы, имеющие простой тип или предопределенные стандартные типы могут содержать только данные (не могут содержать атрибутов и дочерних элементов).

Любой простой тип данных может содержать произвольный набор ограничений, определяемых бизнес-логикой приложения, работающего с данными.

Если простому типу данных присвоено имя, то ссылка на новый нестандартный тип данных может быть использована многократно в пределах данной схемы (аналогично ссылке на предопределенные типы данных).

```
<xs:simpleType name="Код">
     <xs:restriction base="xs:string">
          <xs:length fixed="true" value="4"/>
          <xs:pattern value="[Ю|Ф][0-9]{3}"/>
     </xs:restriction>
</xs:simpleType>
<xs:element name="Код1" type="Код"/>
```

```
<xs:element name="Код2" type="Код"/>
```

В данном примере определен нестандартный тип данных с именем «Код», базирующийся на типе «string». Этот тип данных использован как тип данных для элементов «Код1» и «Код2».

Для описания элементов XML-документа, содержащих дочерние элементы и атрибуты в схеме используется сложный тип данных. Сложный тип данных задается с помощью тега <complexType>.

```
<xs:element name="ИмяЭлемента">
    <xs:complexType>
        Описание сложного типа данных

    </xs:complexType >
</xs:element>
```

При описании сложного типа указывается порядок вхождения дочерних элементов (с помощью специальных тегов - индикаторов порядка, см. таблицу), а также степень кардинальности повторяющихся элементов (с использованием атрибутов minOccurs и maxOccurs).

Атрибут minOccurs определяет минимальную степень кардинальности, т.е. наименьшее возможное количество повторений дочернего элемента. Значение minOccurs равное нулю указывает на необязательность (опциональность элемента). Атрибут maxOccurs определяет максимальную степень кардинальности, или наибольшее количество повторений элемента. Максимальная и минимальная степень кардинальности задаются определенным значением, для maxOccurs может быть указано значение unbounded (элемент встречается любое количество раз).

| Тег индикатора порядка | Описание |
|---|---|
| sequence | Дочерние элементы должны встречаться в указанном порядке. Степень кардинальности определяет, может ли дочерний элемент повторяться. |

| | |
|---|---|
| all | Все дочерние элементы должны присутствовать в XML-документе. Дочерние элементы могут появляться в любом порядке, но должны встречаться только один раз. Нельзя задать значение степени кардинальности maxOccurs и minOccurs отличное от 1. |
| choice | Из всех указанных дочерних элементов должен встречаться только один |

```
<xs:element name="Книга">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Название" type="xs:string" />
         <xs:element name="Автор" type="xs:string" maxOccurs="4"/>
         <xs:element name="Код" type="xs:string"/>
         <xs:element name="Цена" type="xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

В данном примере описан сложный тип данных для элемента «Книга», содержащего дочерние элементы «Название», «Автор», «Код», «Цена». Тег <sequence> является индикатором порядка вхождения дочерних элементов (см. таблицу), атрибут maxOccurs показывает максимально допустимое количество повторений элемента «Автор».

```
<xs:complexType name="Цена">
   <xs:choice>
      <xs:element name="Рубли" type="xs:double" />
      <xs:element name="Доллары" type="xs:double" />
   </xs:choice>
</xs:complexType >
```

Индикатор порядка choice указывает, что элемент этого типа «Цена» может содержать либо элемент «Рубли», либо элемент «Доллары», но не оба.

## ПРИЛОЖЕНИЕ 4

Справочник элементов и атрибутов XSD-схем

| **Элементы XSD-схем** | |
|---|---|
| **Элемент** | **Описание** |
| all | Вложенные элементы могут определяться в произвольном порядке |
| annotation | Родительский элемент элементов-комментариев <appInfo> и <documentation> |
| any | Любые вложенные элементы |
| anyAttribute | Любые атрибуты |
| appInfo | Элемент-коментарий. Задает титул схемы |
| attribute | Атрибут |
| attributeGroup | Группа атрибутов |
| choice | Выбор других элементов. Аналог оператора "\|" в DTD |
| complexContent | Ограничения или расширения модели содержимого сложного типа |
| complexType | Элемент сложного типа |
| documentation | Элемент-комментарий. Предоставляет информацию о схеме |
| element | Элемент |
| extension | Расширения элемента |
| field | Объявление поля. Применяется внутри элемента <unique> для определения полей |
| group | Группа элементов |
| import | Импорт декларации типов из другой схемы |
| include | Включение другой схемы в существующее пространство имен |
| key | Задание элемента или атрибута с ключом, указывающим на другой элемент |
| keyref | Задание элемента или атрибута, на который указывает ключ |
| list | Элемент, который может содержать список значений |
| redefine | Переопределение уже объявленных элементов |
| restriction | Ограничение элемента |
| schema | Корневой элемент схемы |
| selector | Селектор для отбора XML-элементов |
| sequence | Последовательность других элементов. Аналог оператора "," в DTD |
| simpleContent | Модель, содержимое которой представляет только символьные данные |
| simpleType | Элемент простого типа |
| union | Элемент или атрибут, который может иметь множественное значение |
| unique | Элемент или атрибут, который должен иметь уникальное значение |

| Атрибуты XSD-схем | |
|---|---|
| **Атрибут** | **Описание** |
| enumeration | Список значений |
| length | Длина |
| maxLength | Максимальная длина |
| minLength | Минимальная длина |
| maxExclusive | Максимальное значение |
| maxInclusive | Максимальное значение включительно |
| minExclusive | Минимальное значение |
| minInclusive | Минимальное значение включительно |
| fractionDigits | Количество знаков после запятой в дробных числах |
| totalDigits | Количество цифр |
| pattern | Образец (паттерн) содержимого элементов |
| whiteSpace | Количество пробелов в содержимом элементов |
| abstract | Задание элемента абстрактного типа |
| attributeFormDefault | Задание свойств локальных атрибутов как глобальных |
| base | Базовый тип элемента |
| block | Запрещенное выведение ограничением (derivations-by-restriction) |
| blockDefault | Задание начального ограничения block на все определения типов |
| default | Значение элемента или атрибута по умолчанию |
| elementFormDefault | Задание свойств локального элемента как глобально определенного |
| fixed | Фиксированное значение элемента или атрибута |
| form | Локально объявленные элементы определяются в конкретных экземплярах документов |
| itemType | Тип пунктов списка |
| memberTypes | Тип членов, использованных в объединении ( union ) |
| maxOccurs | Максимальное количество вхождений элемента |
| minOccurs | Минимальное количество вхождений элемента |
| mixed | Задание элемента, имеющего смешанный тип |
| name | Название элемента или атрибута |
| namespace | Пространство имен |
| noNamespace | Задание местоположения документа-схемы, |
| SchemaLocation | не имеющего результирующих пространств имен |
| nillable | Определение того, что элемент может иметь пустое значение NULL ( nil ) |
| ref | Задание ссылки на глобально определенный элемент |
| schemaLocation | Определение местоположения схемы |
| substitutionGroup | Определение замены элементов другими элементами |
| targetNamespace | Результирующее пространство имен схемы |
| type | Тип элемента |
| use | Является элемент обязательным или нет |
| value | Значение элемента схемы |
| xsi:nil | Задание реального содержания пустого ( NULL ) элемента XML-документа |
| xsi:schemaLocation | Реальное местоположение элемента в XML-документе |
| xsi:type | Реальный тип элемента в XML-документе |

## ПРИЛОЖЕНИЕ 5

Правила записи регулярных выражений

### Квантификаторы количества: {n}, *, +, ?

| Представление | Число повторений | Пример | Соответствие |
|---|---|---|---|
| **{*n*}** | Ровно *n* раз | colou{3}r | colouuur |
| **{*m*,*n*}** | От *m* до *n* включительно | colou{2,4}r | colouur, colouuur, colouuuur |
| **{*m*,}** | Не менее *m* | colou{2,}r | colouur, colouuur, colouuuur и т. д. |
| **{,*n*}** | Не более *n* | colou{,3}r | color, colour, colouur, colouuur |

| Представление | Число повторений | Эквивалент | Пример | Соответствие |
|---|---|---|---|---|
| * | Ноль или более | {0,} | colou*r | color, colour и т. д. |
| + | Одно или более | {1,} | colou r | colour, colouur и т. д. (но не color) |
| ? | Ноль или одно | {0,1} | colou?r | color, colour |

### Управляющие символы

| Представление | Символ | Расшифровка |
|---|---|---|
| **\t** | Табуляция | Horizontal tabulation |
| **\v** | Вертикальная табуляция | Vertical tabulation |
| **\r** | Возврат каретки | Carriage return |
| **\n** | Перевод строки | Line feed |
| **\f** | Конец страницы | Form feed |
| **\a** | Звонок | Bell character |
| **\e** | Escape-символ | Escape character |
| **\b** | Забой<br><br>Должен находиться внутри квадратных скобок (иначе интерпретируется как граница слова). | Backspace |
| **\cA … \cZ** | Ctrl A … Ctrl Z<br><br>Например, последовательность \cM\cJ соответствует управляющим символам CR LF.<br><br>Эквивалентно \x01 … \x1A. | |
| **\d** | [0-9] | Цифра |

| | | |
|---|---|---|
| **\D** | [^\d] | Любой символ, кроме цифры |
| **\w** | [A-Za-zA-Яа-я0-9_] | Символы, образующие «слово» (буквы, цифры и символ подчёркивания)[1] |
| **\W** | [^\w] | Символы, не образующие «слово» |
| **\s** | [ \t\v\r\n\f] | Пробельный символ |
| **\S** | [^\s] | Непробельный символ |

## Альтернативное сопоставление и группировка выражений

| **Метасимвол** | | **Описание** |
|---|---|---|
| **\|** | Альтернатива | Разделяет альтернативные варианты, часто используется с оператором группировки () |
| **( )** | Группа | Группирует подвыражения для альтернативы, квантификатора или ссылочности |
| **[char]** | Список символов | Обозначает список символов; большинство метасимволов в списке символов представляют собой литеры, за исключением символьных классов и метасимволов ^ и — |
| **[^char]** | Список символов | Список символов, которые не должны присутствовать в строке |