День 2

593. Разработка схем XML-документов с использованием XML Schema

Распределение по дням

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

Стратегии проектирования

При реализации типичных SOA-проектов, как правило, создается несколько XML-схем. В этих случаях проектировщик XML-схем должен решить следующий вопрос:

- необходимо ли всем XML-схемам в проекте присваивать различные значения targetNamespace?
- или нужно использовать единый targetNamespace для всех?
- и можно ли некоторым XML-схемам не присваивать никакого targetNamespace?

Стратегии проектирования

Выделяют три проектных подхода при работе с несколькими XML-схемами:

- **гетерогенное пространство имен** - каждой XML-схеме присваивается свой targetNamespace;
- **гомогенное пространство имен** - всем XML-схемам присваивается единый targetNamespace;
- **пространство 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).

Гетерогенное пространство имен

Product.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.product.org"
            xmlns="http://www.product.org"
            elementFormDefault="unqualified">
            xmlns:per="http://www.person.org"
            xmlns:pro="http://www.product.org">
    <xsd:complexType name="ProductType">
        <xsd:sequence>
           <xsd:element name="Type" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Person.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.person.org"
            xmlns="http://www.person.org"
            elementFormDefault="unqualified">
    <xsd:complexType name="PersonType">
        <xsd:sequence>
           <xsd:element name="Name" type="xsd:string"/>
           <xsd:element name="SSN" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Company.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.company.org"
            elementFormDefault="unqualified"
            xmlns:per="http://www.person.org"
            xmlns:pro="http://www.product.org">
    <xsd:import namespace="http://www.person.org"
                schemaLocation="Person.xsd"/>
    <xsd:import namespace="http://www.product.org"
                schemaLocation="Product.xsd"/>
    <xsd:element name="Company">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Person" type="per:PersonType"
                             maxOccurs="unbounded"/>
                <xsd:element name="Product" type="pro:ProductType"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```


XML

Гомогенное пространство имен

Product.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.product.org"
            elementFormDefault="qualified">
    <xsd:complexType name="ProductType">
        <xsd:sequence>
           <xsd:element name="Type" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Person.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.person.org"
            elementFormDefault="qualified">
    <xsd:complexType name="PersonType">
        <xsd:sequence>
           <xsd:element name="Name" type="xsd:string"/>
           <xsd:element name="SSN" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Company.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.company.org"
            elementFormDefault="qualified">
    <xsd:include schemaLocation="Person.xsd"/>
    <xsd:include schemaLocation="Product.xsd"/>
    <xsd:element name="Company">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Person" type="PersonType"
                             maxOccurs="unbounded"/>
                <xsd:element name="Product" type="ProductType"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

Пространство имен типа "хамелеон"

Product.xsd (нет targetNamespace)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
    <xsd:complexType name="ProductType">
        <xsd:sequence>
           <xsd:element name="Type" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Person.xsd (нет targetNamespace)

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            elementFormDefault="qualified">
    <xsd:complexType name="PersonType">
        <xsd:sequence>
           <xsd:element name="Name" type="xsd:string"/>
           <xsd:element name="SSN" type="xsd:string"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:schema>
```

Company.xsd

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.company.org"
            elementFormDefault="qualified">
    <xsd:include schemaLocation="Person.xsd"/>
    <xsd:include schemaLocation="Product.xsd"/>
    <xsd:element name="Company">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Person" type="PersonType"
                             maxOccurs="unbounded"/>
                <xsd:element name="Product" type="ProductType"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>>
```

Разница в использовании

Company.xml (для версии с гетерогенным пространством имен в targetNamespace)

```
<?xml version="1.0"?>
<Company xmlns="http://www.company.org"
         xmlns:pro="http://www.product.org"
         xmlns:per="http://www.person.org"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.company.org/Company.xsd">
        <Person>
                <per:Name>John Doe</per:Name>
                <per:SSN>123-45-6789</per:SSN>
        </Person>
        <Product>
                <pro:Type>Widget</pro:Type>
        </Product>
</Company>
```

Обратите внимание на следующее:

- необходимо иметь декларацию namespace для кадого пространства имен;
- все элементы должны быть уникально квалифицированы (явно или через пространство имен "по-умолчанию").

Разница в использовании

Company.xml (для версии с гомогенным пространством имен в targetNamespace)

```
<?xml version="1.0"?>
<Company xmlns="http://www.company.org"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.company.org/Company.xsd">
        <Person>
                <Name>John Doe</Name>
                <SSN>123-45-6789</SSN>
        </Person>
        <Product>
                <Type>Widget</Type>
        </Product>
</Company>
```

Поскольку все схемы принадлежат одному пространству имен, то в XML-документах для данной ситуации можно воспользоваться преимуществом использования пространства имен "по-умолчанию".

Разница в использовании

Company.xml (для версии с пространством имен типа "хамелеон" в targetNamespace)

```
<?xml version="1.0"?>
<Company xmlns="http://www.company.org"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.company.org/Company.xsd">
        <Person>
                <Name>John Doe</Name>
                <SSN>123-45-6789</SSN>
        </Person>
        <Product>
                <Type>Widget</Type>
        </Product>
</Company>
```

Обе XML-схемы без определения targetNamespace приняли targetNamespace XML-схемы Company.xsd (подобно хамелеон-эффекту). Таким образом, все компоненты принадлежат одному targetNamespace, и в XML-документах для данной ситуации также можно воспользоваться преимуществом использования пространства имен "по-умолчанию".

redefine

<redefine> - применяется только в гомогенном пространстве имен и в пространстве имен типа "хамелеон"

Элемент <redefine> используется в XML-схемах для получения доступа к компонентам в других XML-схемах и одновременно дает возможность внести какое-то число (ноль или более) изменений 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.

redefine

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            targetNamespace="http://www.company.org"
            xmlns="http://www.company.org"
            elementFormDefault="qualified">
    <xsd:include schemaLocation="Person.xsd"/>
    <xsd:redefine schemaLocation="Product.xsd">
        <xsd:complexType name="ProductType">
            <xsd:complexContent>
                <xsd:extension base="ProductType">
                    <xsd:sequence>
                        <xsd:element name="ID" type="xsd:ID"/>
                    </xsd:sequence>
                </xsd:extension>
            </xsd:complexContent>
        </xsd:complexType>
    </xsd:redefine>
    <xsd:element name="Company">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Person" type="PersonType"
                             maxOccurs="unbounded"/>
                <xsd:element name="Product" type="ProductType"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

```
<?xml version="1.0"?>
<Company xmlns="http://www.company.org"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://www.company.org/Company.xsd">
        <Person>
                <Name>John Doe</Name>
                <SSN>123-45-6789</SSN>
        </Person>
        <Product>
                <Type>Widget</Type>
                <ID>1001-01-00</ID>
        </Product>
</Company>
```

Коллизия имен

Когда главная XML-схема использует хамелеон-комоненты, то эти компоненты становятся частью пространства имен (указанного в targetNamespace) главной XML-схемы, так как будто проектировщик XML-схемы использовал in-line декларацию элементов и типов. Если главная схема включает несколько XML-схем без указания пространства имен, то существует шанс, что возникнет коллизия имен. Таким образом, главная XML-схема не сможет использовать некоторые компоненты вспомогательных XML-схем, поскольку для них имеет место коллизия имен с элементами из других вспомогательных XML-схем.

Предположим, существует две XML-схемы без указания targetNamespace:

1.xsd: A, B

2.xsd: A, C

Коллизия имен

XML-схема 1.xsd определяет элементы A и B без указания пространства имен.
XML-схема 2.xsd определяет элементы A и C без указания пространства имен.
Теперь, если XML-схема 3.xsd включает в себя (<include>) две указанные XML-схемы без указания пространства имен, возникает коллизия имен для элемента A, поскольку он объявлен два раза:
3.xsd targetNamespace="http://www.example.org"
<include schemaLocation="1.xsd"/>
<include schemaLocation="2.xsd"/>

Ошибкой не является существование определения двух элементов с одинаковым именем, если они принадлежат к одному типу. Если же они принадлежат различным типам, то это ошибка, и имеет место коллизия имен.

Устранение коллизий

Стандартным механизмом исключения коллизий имен как раз и является применение пространств имен.

Существует очень простое решение данной проблемы коллизии имен для связи типа "хамелеон": для каждой включаемой XML-схемы без указания пространства имен создается вспомогательная проксирующая XML-схема, в которой декларировано пространство имен и которая сама уже включает (<include>) вспомогательные XML-схемы без указания пространств имен. Затем главная схема просто импортирует (<import>) все проксирующие XML-схемы.

Устранение коллизий

```
<!-- Поместим "хамелеон"-компоненты XML-схемы 1.xsd
в пространство имен, указанное в targetNamespace, данной проксирующей XML-схемы -->
1-proxy.xsd
targetNamespace="http://www.1-proxy.org"
<xsd:include schemaLocation="1.xsd"/>

<!-- Поместим "хамелеон"-компоненты XML-схемы 2.xsd
в пространство имен, указанное в targetNamespace, данной проксирующей XML-схемы -->
2-proxy.xsd
targetNamespace="http://www.2-proxy.org"
<xsd:include schemaLocation="2.xsd"/>

<!-- Теперь импортируем проксирующие XML-схемы в главную схему -->
main.xsd
targetNamespace="http://www.main.org"
<xsd:import namespace="http://www.1-proxy.org"
        schemaLocation="1-proxy.xsd"/>
<xsd:import namespace="http://www.2-proxy.org"
        schemaLocation="2-proxy.xsd"/>
```

Таким образом, данный проектный подход регламентирует трехступенчатый процесс:

- создать хамелеон-схемы;
- создать проксирующие XML-схемы для каждой хамелеон-схемы;
- импортировать (<import>) проксирующие XML-схемы в главную.

Рекомендации

Используйте гомогенное пространство имен:

- если все XML-схемы концептуально соотносятся друг с другом;
- если нет необходимости визуально идентифицировать в XML-документе принадлежность элементов и атрибутов той или иной XML-схеме. При данном подходе все компоненты принадлежат одному пространству имен и, таким образом, теряется возможность идентифицировать в XML-документе, что "элемент A определен в схеме X". Часто это нормально, что проектировщик не желает категоризовывать раздельно элементы или атрибуты. В этом случае гомогенное пространство имен вполне подходит.

Рекомендации

Используйте гетерогенное пространство имен:
• когда имеется несколько элементов с одинаковым именем (с целью предотвратить коллизию имен);
• если есть необходимость визуально идентифицировать в XML-документе принадлежность элементов и атрибутов той или иной XML-схеме. При данном подходе компоненты принадлежат различным пространствам имен, и, таким образом, существует возможность идентифицировать в XML-документе, что "элемент A определен в схеме X".

RELAX NG

RELAX NG (REgular LAnguage for XML Next Generation) — один из языков описания структуры XML-документа. Являясь сама по себе XML-документом, схема в этом формате может быть записана с использованием альтернативного, более компактного синтаксиса. В сравнении с другими языками схем, RELAX NG относительно прост. RELAX NG была разработан в OASIS и впервые опубликован в 2003. Файлы, содержащие схемы RELAX NG, обычно имеют расширение ".rng", а в компактном синтаксисе — ".rnc".

Пример

```
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
   <start>
      <element name="book">
         <oneOrMore>
            <ref name="page"/>
         </oneOrMore>
      </element>
   </start>
   <define name="page">
      <element name="page">
         <text/>
      </element>
   </define>
</grammar>
```

```
element book
{
    element page { text }+
}
```

Или, с *именованными паттернами*:

```
start = element book { page+ }
page = element page { text }
```

Синтаксис

<attribute name="id"/>

attribute id { text }

<element name="name" />

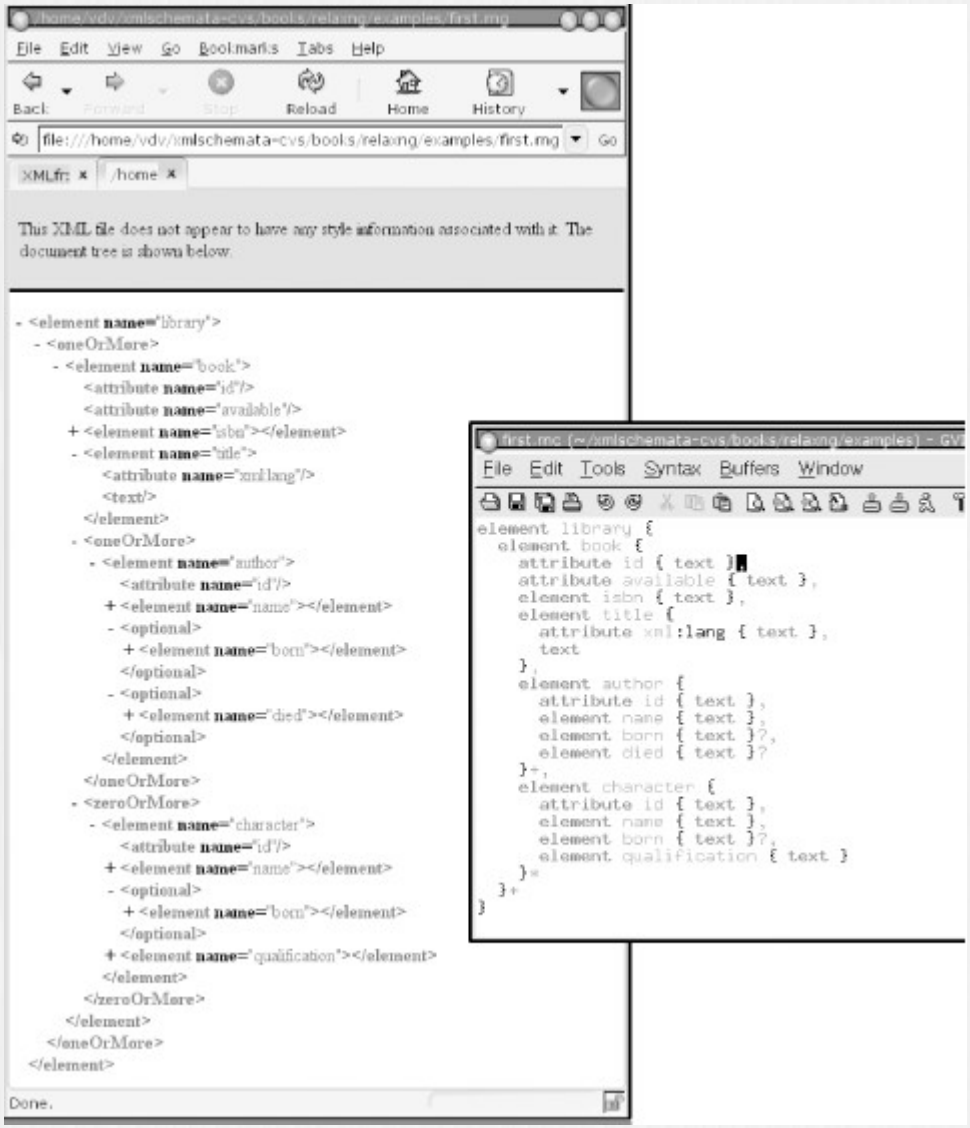
element name { text }

element author { attribute id { text }, element name { text }, element born { text }, element died { text } }

Пример

```
element library {
   element book {
      attribute id { text },
      attribute available { text },
      element isbn { text },
      element title {
        attribute xml:lang { text },
        text
      },
      element author {
        attribute id { text },
        element name { text },
        element born { text }?,
        element died { text }?
      }+,
      element character {
        attribute id { text },
        element name { text },
        element born { text }?,
        element qualification { text }
      }*
   }+
}
```

XML Schema и RELAX NG



XML

Редакторы XML схем, документов

Name	Version	Software license	Active	Type	Windows	Mac OS X	Linux	Textual editor?	Graphical editor?	WYSIWYG editors?	Syntax highlighting for (XML,XSD,XSL,XSLT,...)	Tag Folding	Intellisense/Autocompletion (using schema.xsd for text)
CAM editor	2.2 (2012)	OSL	Yes	Cross-platform	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No
EditiX XML Editor	2012	Proprietary	Yes	Standalone	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
emacs/ xml.el[1],nxml-el [2]	current/ 2003?	GNU GPL	No	Emacs modes	Yes	Yes	Yes	Yes	Yes	No	?	?	?
Jedit XML Plugin	2.6.1 (2011)	GNU GPL	Yes	Plugin	Yes	Yes	Yes	?	?	?	?	?	?
Liquid XML Studio	10.0 (2012)	Proprietary	Yes	Standalone	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Oxygen XML Editor	13.2 (2012)	Proprietary	Yes	Standalone[3]	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Quark XML Author	3	Proprietary	Yes	Standalone	Yes	No	No	Yes	Yes	No	?	?	?
Rinzo	1.1.0	GNU LGPL	Yes	Eclipse Plugin	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes[4]
Serna XML Editor	4.3 (2011)	GNU GPL, Proprietary	Yes	Standalone	Yes	Yes	Yes	Yes	Yes	Yes	?	?	?
Stylus Studio XML IDE	X14 (2012)	Proprietary	Yes	Standalone	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes
Xerlin	1.3 (2005)	Open source	No	Standalone	Yes	Yes	Yes	?	?	?	?	?	?
XMetaL	7.0	Proprietary	Yes	Standalone	Yes	No	No	Yes	Yes	Yes	?	?	?
XML Notepad	2.5 (2007)	Microsoft Public License	No	Standalone	Yes	No	No	Yes	Yes	?	?	?	?
XMLSpy	2012	Proprietary	Yes	Standalone Eclipse Plugin	Yes	No[5]	Yes[5][6]	Yes	Yes	Yes	Yes	Yes	Yes
XMLBlueprint XML editor	2011	Proprietary	Yes	Standalone	Yes	No	No	Yes	No	No	Yes	Yes	Yes
XML Copy Editor	1.2.0.6	GNU GPL	Yes	Standalone	Yes	Yes	Yes	Yes	?	?	?	?	?

Excel и XML

Основной процесс использования XML-данных в Excel:

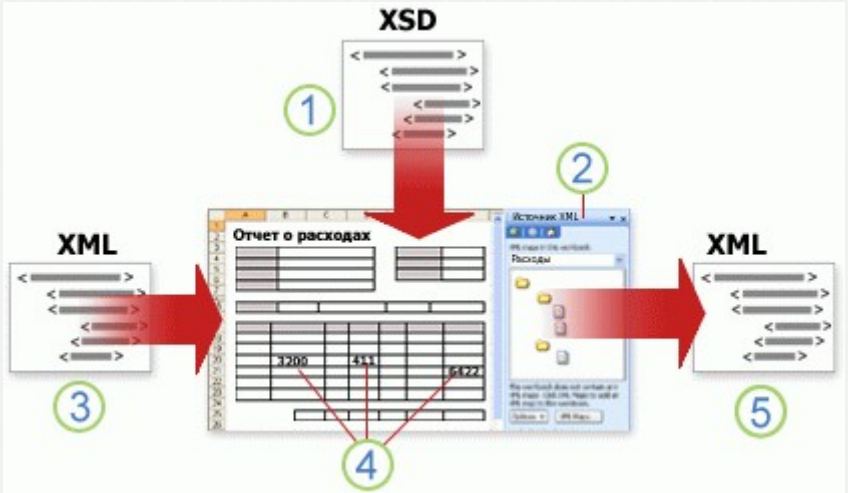
1.Добавление в книгу файла XML-схемы.

2. Сопоставление элементов XML-схемы с отдельными ячейками или XML-таблицами.

3. Импорт файла XML-данных (XML) и привязка XML-элементов к сопоставленным ячейкам.

4. Ввод данных, перемещение сопоставленных ячеек и улучшение функциональных возможностей Excel при сохранении XML-структуры и определений.

5. Экспорт обработанных данных из сопоставленных ячеек в файл XML-данных.



Работа с картами XML

Можно создать или открыть книгу в Excel, вложить файл XML-схемы (XSD) в книгу и использовать область задач Источник XML для сопоставления XML-элементов схемы с отдельными ячейками или таблицами. Реализовав такое сопоставление, можно импортировать и экспортировать XML-данные в ячейки таблицы и из них соответственно.

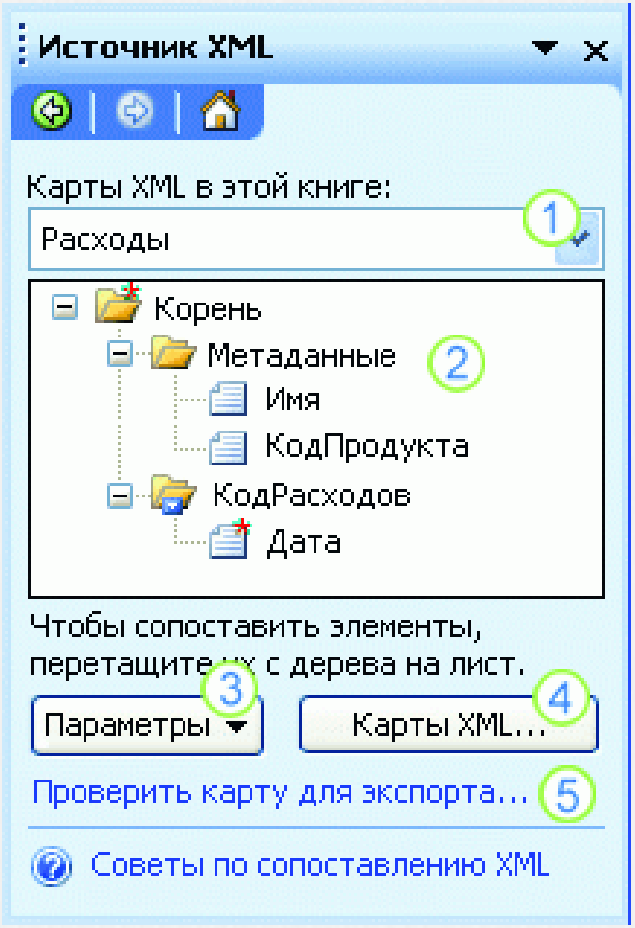
При добавлении файла схемы XML (XSD) в книгу создается карта XML. Карты XML используются для создания сопоставленных ячеек и управления взаимосвязью между этими сопоставленными ячейками и отдельными элементами XML-схемы. Кроме того, эти карты используются для привязки содержимого сопоставленных ячеек к элементам схемы при импорте или экспорте файлов XML-данных (XML).

Работа с картами XML

Необходимо знать следующие правила использования карт XML.

- Книга может содержать одну или несколько карт XML.
- Одновременно можно сопоставить только один элемент с одним местоположением в книге.
- Каждая карта XML независима от других, даже если несколько карт XML в одной книге относятся к одной и той же схеме.
- Карта XML может содержать только один корневой элемент. При добавлении схемы, определяющей более одного корневого элемента, появится запрос на выбор корневого элемента, который будет использоваться в новой карте XML.

Использование области задач источника XML



Область задач Источник XML используется для управления картами XML. Чтобы открыть ее, необходимо на вкладке Разработчик в группе XML щелкнуть Источник.

Типы элементов и их значки

ТИП ЭЛЕМЕНТА	ЗНАЧОК
Родительский элемент	
Обязательный родительский элемент	
Повторяющийся родительский элемент	
Обязательный повторяющийся родительский элемент	
Дочерний элемент	
Обязательный дочерний элемент	
Повторяющийся дочерний элемент	
Обязательный повторяющийся дочерний элемент	
Атрибут	
Обязательный атрибут	
Простое содержимое в сложной структуре	
Обязательное простое содержимое в сложной структуре	

Работа с отдельно сопоставленными ячейками

Отдельно сопоставленная ячейка — это ячейка, сопоставленная с неповторяющимся XML-элементом. Такую ячейку можно создать, переместив неповторяющийся XML-элемент из области задач Источник XML в отдельную ячейку книги.

Название перемещаемого на лист неповторяющегося XML-элемента можно назначить заголовком сверху или слева от отдельно сопоставленной ячейки с помощью смарт-тега. В качестве заголовка можно также использовать текущее значение ячейки.

В отдельно сопоставленной ячейке можно также использовать формулу, если ячейка сопоставлена с XML-элементом, имеющим тип данных определения XML-схемы (XSD), интерпретируемым Microsoft Excel как число, дата или время.

Работа с повторяющимися ячейками в XML-таблицах

По внешнему виду и функциональным возможностям XML-таблицы подобны таблицам Microsoft Excel. XML-таблицы являются таблицами Microsoft Excel, сопоставленными с одним или несколькими повторяющимися XML-элементами. Каждый столбец XML-таблицы соответствует XML-элементу.

При работе с XML-таблицами полезны два параметра, доступ к которым можно получить с помощью кнопки Параметры в области задач Источник XML:

Автоматически объединять элементы при сопоставлении. При выборе этого параметра в Excel создается одна XML-таблица из нескольких полей, переносимых на лист.

Данные с заголовками. При установке этого флажка существующие данные заголовков будут использоваться в качестве названий столбцов для повторяющихся элементов, сопоставляемых на листе.

Импорт XML-данных

В диалоговом окне Свойства карты XML (щелкните Свойства карты в группе XML на вкладке Разработчик) существуют три параметра, установленные по умолчанию. С их помощью можно управлять поведением привязок XML-данных.

- **Проверять данные на соответствие схеме при импорте и экспорте.** Включите этот параметр, если необходимо, чтобы импортируемые XML-данные соответствовали XML-схеме.
- **Заменять существующие данные новыми.** Указывает, следует ли заменять данные при импорте. Включите этот параметр, если необходимо заменять текущие данные новыми, например, если в новом файле XML-данных содержатся более новые данные.
- **Добавлять новые данные в существующие XML-таблицы.** Указывает, следует ли добавлять содержимое источника данных к существующим данным на листе.

Экспорт XML-данных

При экспорте данных в Excel применяются следующие правила:

- Пустые элементы не создаются, если пустые ячейки существуют для дополнительного элемента, однако пустые элементы создаются, если пустые ячейки существуют для требуемого элемента.
- Запись данных производится с помощью кодировки UTF-8.
- Все пространства имен определяются в корневом XML-элементе.
- Существующие префиксы пространства имен переопределяются. По умолчанию пространству имен назначается префикс ns0, следующие ns1 и т.д.
- Узлы приложений не сохраняются.

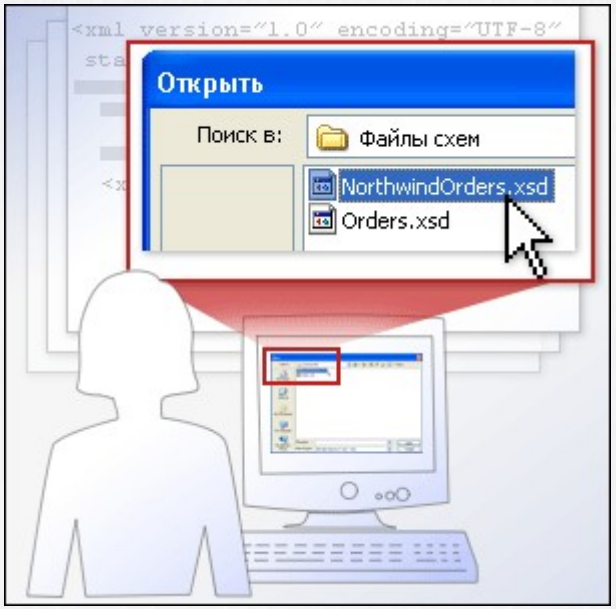
В диалоговом окне Свойства карты XML (щелкните Свойства карты в группе XML на вкладке Разработчик) используйте параметр **Проверять данные на соответствие схеме при импорте и экспорте** (по умолчанию включен), чтобы указать, должны ли данные проверяться по карте XML при экспорте данных. Включите этот параметр, если следует убедиться, что все экспортируемые XML-данные соответствуют XML-схеме.

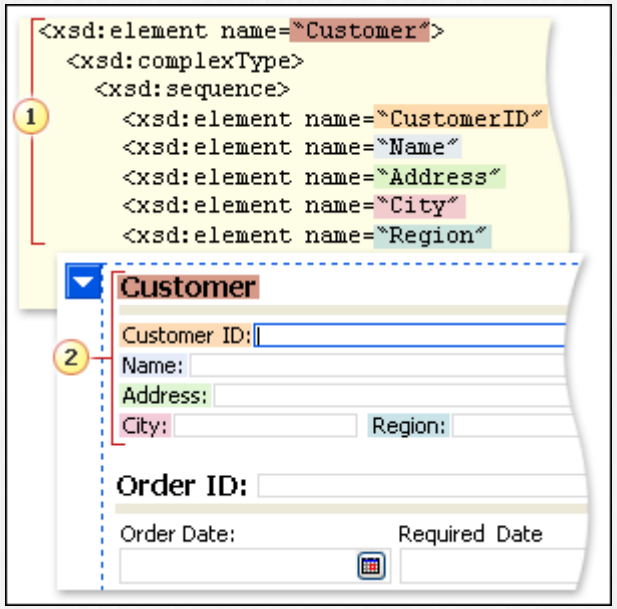
Infopath и XML

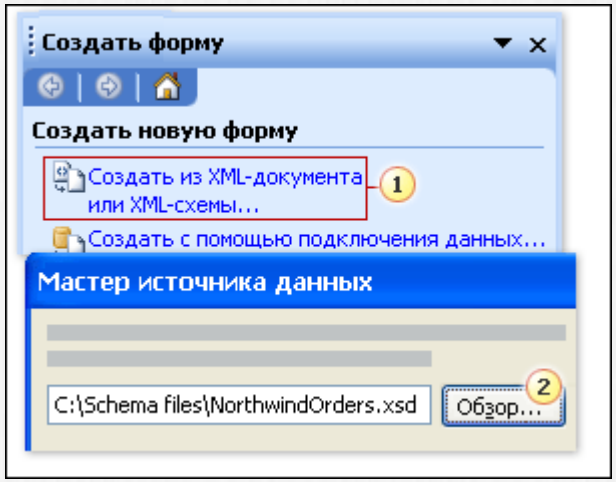
Microsoft InfoPath (полное название Microsoft Office InfoPath) — приложение, используемое для разработки форм ввода данных на основе XML. Впервые это приложение появилось как часть Microsoft Office 2003 в конце 2003 года, а затем было выпущено в составе Microsoft Office 2007. При разработке приложение первоначально носило кодовое имя «NetDocs», затем «XDocs».

Основная возможность InfoPath — возможность создавать, просматривать и редактировать документы, поддерживающие некоторую заданную пользователем XML-схему. Для получения и изменения данных можно использовать соединение с внешними системами — базами данных Access, MS SQL, веб-сервисами.

Infopath

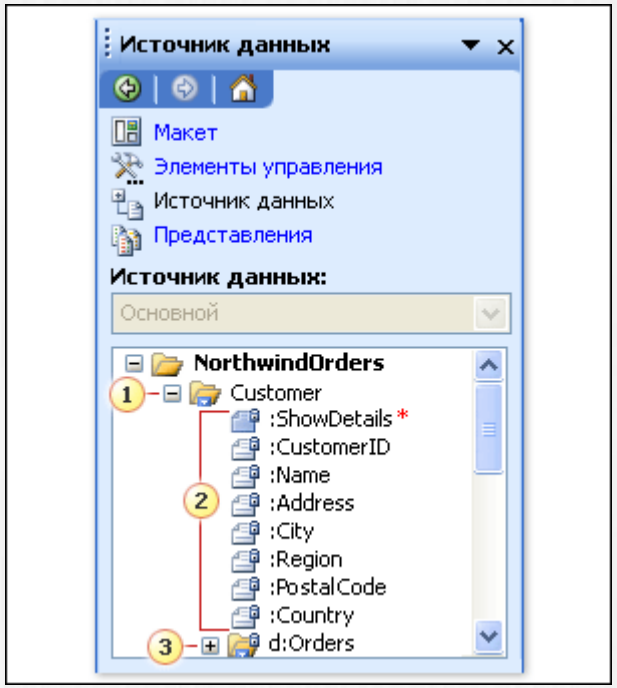






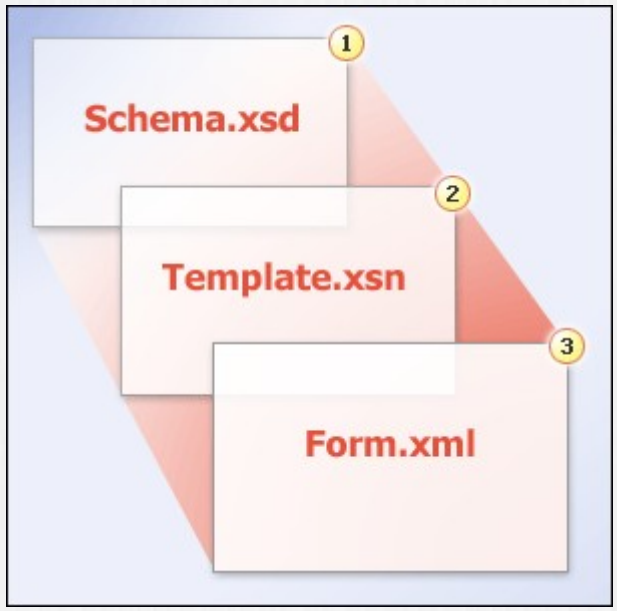
XML

Infopath



После выбора схемы, на основе которой будет создаваться форма, можно просмотреть ее структуру и элементы в области задач Источник данных. Элементы схемы отображаются в виде полей и групп полей, как показано на рисунке. В области задач можно увидеть, какие поля относятся к каждой группе и в каком порядке они следуют.

Infopath



Связь между схемой XML, шаблоном формы и формой, созданной с использованием этого шаблона.

1. Схема является основой шаблона формы.
2. Разработанный шаблон формы используется для создания формы, предназначенной для заполнения.
3. Форма сохраняется вместе с введенными данными, отдельно от шаблона формы.

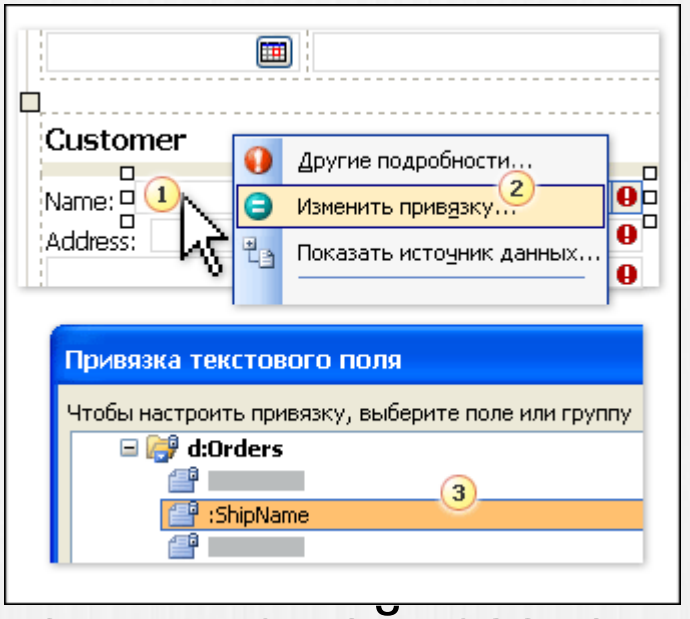
Infopath

На основе сведений о типах данных из схемы XML в InfoPath особым полям автоматически назначаются определенные типы элементов управления. Например, при перетаскивании поля string в форму, InfoPath автоматически преобразует его в элемент управления "Текстовое поле", поскольку для этого элемента схемы указан текстовый тип данных.
Иногда для некоторых данных подходит несколько элементов управления. В таких случаях в InfoPath предоставляется несколько вариантов для выбора. Например, определяются группы, которые могут повторяться, и предлагаются соответствующие элементы управления.

Infopath

Когда указатель мыши помещен на элемент управления, в правом верхнем углу появляется зеленый значок привязки, указывающий, к какому элементу привязан элемент управления. В данном случае элемент управления Повторяющийся раздел привязан к группе Customer, и зеленый значок подтверждает, что привязка выполнена правильно.

Infopath



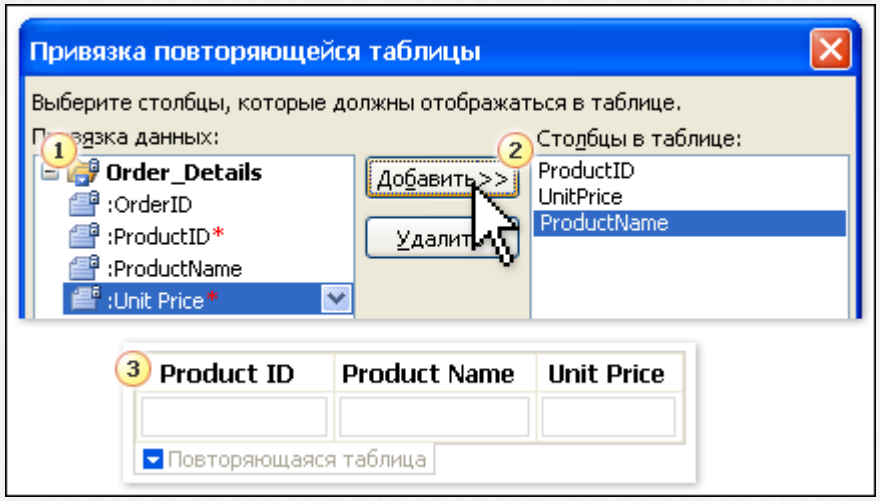
Если таблица макета Customer копируется в другой раздел формы, появляется значок в виде восклицательного знака в красном кружке, показывающий, что данный элемент управления не привязан к структуре схемы. Для использования этой таблицы макета достаточно переопределить привязку элементов управления (и ввести новый заголовок):

1. Чтобы изменить привязку элемента управления Name, щелкните его.
2. В контекстном меню выберите команду Изменить привязку.
3. В появившемся окне привязки выберите поле, к которому требуется привязать данный элемент управления.

Infopath

Диалоговое окно Привязка повторяющейся таблицы позволяет быстро привязать столбцы таблицы к полям данных.

1. Выберите все поля, которые должны появиться в столбце таблицы.
2. Добавьте все выбранные поля в список Столбцы таблицы.
3. Все элементы управления в столбце таблицы, привязанные к выбранным полям, добавляются в форму одновременно. Они размещаются в том порядке, в каком были выбраны.



XML

Java и XML Shema

До недавнего времени API (прикладной интерфейс), через который приложения выполняли валидацию, определялся языком описания схем и используемым парсером. Работа с DTD и XSD осуществлялась через опции интерфейсов SAX (Simple API for XML), DOM (Document Object Model) и JAXP (Java API for XML Processing). Для использования RELAX NG была необходима специальная библиотека и соответствующий API, а для Schematron – например, TrAX (Transformations API for XML). Другие языки описания схем также требовали от разработчиков изучения специальных API, хотя все они выполняли аналогичные действия.

В Java 5 появился пакет javax.xml.validation, представляющий собой независящий от языка схем интерфейс использования механизма валидации. Этот пакет также был доступен начиная с Java 1.3, но для этого необходимо было отдельно установить JAXP 1.3. Его реализация является одним из компонентов библиотеки Xerces 2.8.

Java. Валидация

В пакете javax.xml.validation для валидации документов используются три класса: SchemaFactory, Schema и Validator. Кроме того, этот пакет активно использует интерфейс javax.xml.transform.Source из TrAX для представления документов XML. Если не вдаваться в детали, то назначение этих классов таково: SchemaFactory читает описание схемы, которая часто представляет собой документ XML, создавая на его основе экземпляр типа Schema, который в свою очередь создает валидатор (объект типа Validator). Валидатор выполняет проверку документа XML, представленного в виде экземпляра Source.

см. пример XMLSchema1

Java. Валидация

Процесс валидации всегда состоит из следующих пяти этапов.

- Загрузка фабрики для выбранного языка описания схем.
- Компиляция схемы по ее описанию.
- Создание валидатора на основе скомпилированной схемы.
- Создание объекта типа Source для представления валидируемого документа. Как правило, в качестве реализации такого объекта проще всего использовать класс StreamSource.
- Валидация документа. Если документ оказывается некорректным, метод validate() выбрасывает исключение типа SAXException, а в противном случае он просто завершает выполнение.

Java. Обработка ошибок

По умолчанию валидатор выбрасывает исключение типа SAXException в случае некорректного содержимого документа, а в противном случае он просто тихо завершает работу. Однако существует возможность получения более детальной информации о проблемах в документе через классы, реализующие ErrorHandler в SAX. Например, допустим, что вы хотите заносить в журнал все ошибки валидации, но при этом очередная ошибка не должна останавливать процесс проверки.

см. пример XMLSchema1, класс ForgivingErrorHandler

Java. Изменение документа

Возможности схем не обязательно ограничиваются валидацией документов. Некоторые из них могут не только проверять корректность содержимого документа, но также добавлять в него дополнительную информацию, например, указывать значения по умолчанию для атрибутов элементов. Кроме того, они могут указывать типы данных, например int или gYear, для атрибутов и элементов. При этом валидаторы записывают подобные изменения в объект типа javax.xml.transform.Result, который в этих случаях должен передаваться вторым параметром в метод validate.

Java. Информация о типах

Понятие типа является одним из центральных в языке XML Schema. Элементы и атрибуты имеют свои типы, например, int, double, date, duration, person, PhoneNumber или какие-либо еще. API валидации в Java предоставляет возможности определения типов, хотя эта функциональность, как ни странно, не зависит от остальных компонентов инфраструктуры.

Типы представляются в виде экземпляров org.w3c.dom.TypeInfo. Этот интерфейс позволяет определить локальное имя типа, URI его пространства имен, а также то, является ли он расширением некоего базового типа. Все остальные манипуляции с типами остаются на усмотрение вашего приложения. API не предоставляет никакой информации о семантике типов или об их преобразованиях в Java-типы, например, double или java.util.Date.

Java. Информация о типах

Для получения экземпляров TypeInfo следует запросить у экземпляра Schema специальный объект-обработчик ValidatorHandler, реализующий SAX-интерфейс ContentHandler, а не Validator. Его следует передавать SAX-парсеру при разборе документа.

Вы можете зарегистрировать собственный класс ContentHandler в объекте ValidatorHandler (а не в парсере). В этом случае ValidatorHandler будет делегировать события вашему объекту.

Класс ValidatorHandler предоставляет доступ к объекту TypeInfoProvider, который может использоваться вашим обработчиком ContentHandler для определения типа текущего элемента или его атрибутов. При помощи этого класса можно также определить, является ли атрибут идентификатором, был он явно указан в документе или появился в процессе валидации по схеме

Источники

1. Разработка формы на основе XML (http://office.microsoft.com/ru-ru/training/RZ001122374.aspx)
2. Обзор XML в Excel (http://office.microsoft.com/ru-ru/excel-help/HA010206396.aspx)
3. Практика использования пространств имен XML в проектах, содержащих несколько XML-схем (http://www.interface.ru/home.asp?artId=21058)
4. RELAX NG (http://wiki.telekomza.ru/wiki/RELAX_NG)
5. RELAX NG (http://books.xmlschemata.org/relaxng/page2.html)