Java Advanced

Введение в XML

Часть 1

XML

eXtensible Markup Language

- XML – текстовый язык хранения структурированных данных
- Предшественники XML
  - Standard Generalized Markup Language (SGML)
  - Hyper Text Markup Language (HTML)

Составляющие XML-документа

- Элементы (element) – задают структуру элемента
- Атрибуты (attribute) – дополняют информацию об элементе
- Символьные данные (character data) – текст внутри элемента
- Указания по обработке (processing instruction) – применяются разборщиками и другими программами
- Комментарии (comment) – игнорируются

Элемент

- Структура
  - Имя
  - Дети
  - Атрибуты
- Синтаксис

  <ИмяЭлемента Аттрибуты>

  Дети

  </ИмяЭлемента>
- Сокращенный

  <ИмяЭлемента Аттрибуты/>

Атрибут

- Структура
  - Имя
  - Значение
- Синтаксис

  имя = "значение"

  или

  имя = 'значение'

Пример: элементы и атрибуты

- Описание книги

```
<book
    caption        = "Рефакторинг"
    isbn           = "5-93286-045-6"
    publisher      = "Символ-Плюс"
    pages          = "430"
>
    <author name='Мартин' last-name='Фаулер'/>
</book>
```

XML-идентификаторы

- Первый символ
  - Буква, _ или :
- Последующие символы
  - Первый символ, цифра, - или .
- Примеры
  - **hello**
  - **HelloWorld**
  - **HelloWorld156Times**
  - **hello-world**
  - **hello.world**

Символьные данные

- Простые символы
  - Пример: Привет!
- Именованные ссылки
  - &название;
  - Пример: Пусть a < b и b < c, тогда a < c
- Указание кода символа
  - &#НомерСимвола;
  - &#xШестнадцатеричныйНомерСимвола;
  - Пример: A A

Именованные ссылки

- Значения по умолчанию

&amp;	&	ampersand
&lt;	<	less than
&gt;	>	greater than
&quot;	"	quotes
&apos;	'	apostrophe

Пробелы и переводы строк

- Пробельные символы
  - #x20 — Пробел
  - #x9 — Табуляция
  - #xA — Перевод строки
  - Переводы строки
- Переводы строки
  - #xD — возврат каретки
  - #x85 — перевод строки (IBM)
  - #x2028 — перевод строки (Unicode)
  - #xD #xA — перевод строки (DOS)
  - #xD #x85

Блоки символьных данных

- Синтаксис
  - <![CDATA[символьные данные]]>
- Примеры
  - <![CDATA[Внутри блока символьных данных спец символы можно писать непосредственно: ? < > ' "]]>
  - <![CDATA[Но иногда приходится делать так:]]]]><![CDATA[>]]>

Указания по обработке

- Структура
  - Имя
  - Значение
- Синтаксис
  - **<?**имя значение**?>**

Заголовок XML-файла

- Синтаксис
  - <?xml version="версия" encoding="кодировка"?>
- Примеры

  Версия 1.1, кодировка 1251 (Windows Russian)

  <?xml version="1.1" encoding="WINDOWS-1251"?>

  Версия 1.0, кодировка 866 (DOS Russian)

  <?xml encoding="Cp866"?>

  Версия 1.0, кодировка – автоопределение (UTF-8, UTF-16)

  <?xml version="1.0"?>

Комментарии

- Синтаксис
  - **<!--** комментарий **-->**
- Примеры

<!-- простой комментарий **-->**

<!--

многострочный

комментарий

-->

XML-документ

- Пролог
  - Заголовок
  - Тип документа
  - Комментарии и указания
- Элемент
- Эпилог
  - Комментарии и указания

Часть 2

Пространства имен

Пространства имен

- Позволяют одновременно использовать одинаковые имена, придуманные разными людьми
- Пространство имен идентифицируется URI

Указание пространства имен

- Структура
  - Полное имя (qualified name)
  - Пространство имен (namespace)
  - Локальное имя (local name)
  - Префикс (prefix)
- Имя имеет вид
  - префикс**:**локальноеИмя
- Пространства имен связываются с префиксами с помощью атрибутов вида
  - **xmlns:префикс = "**пространство имен**"**

Пример: пространства имен

- Описание книги

```
<library:book
    xmlns:library  = "http://example.com/MyLibrary"
    xmlns:isbn     = "http://example.com/isbn"
    xmlns:issn     = "http://example.com/issn"
    isbn:number  = "5398866"
    issn:number  = "unknown"
    …
>
    <library:author library:name="Мартин" .../>
   …
</library:book>
```

Область действия префикса

- От элемента для которого определено отображение префикса в низ по дереву, до элементов для которых указано новое отображение этих префиксов
- Действие префикса можно отменить, указав отображение на пустую строку

Пространство имен по умолчанию

- Применяется для элементов для которых не указано пространство имен
- Объявление
  - **xmlns="**пространство имен**"**

Пример: пространства имен по умолчанию

- Описание книги

```
<book
    xmlns              = " http://example.com/MyLibrary"
    xmlns:library      = "http://example.com/MyLibrary"
    xmlns:isbn         = "http://example.com/isbn"
    xmlns:issn         = "http://example.com/issn"
    isbn:number        = "5398866"
    issn:number        = "unknown«
    …
>
     <author library:name="Мартин" .../>
   …
</book>
```

Часть 3

SAX

Simple API for XML

- Представляет XML-документ в виде последовательности событий
- Пакеты
  - org.xml.sax – модель SAX
  - java.xml.parsers – разборщики

Разбор XML

- Интерфейс XMLReader
- Методы
  - parse(InputSource) – разобрать XML-документ
  - setContentHandler(ContentHandler) – устанавливает приемник событий
  - setErrorHandler(ErrorHandler) – установить обработчик ошибок
  - setFeature(name, value) – установить настройку
  - setProperty(name, value) – установить свойство

Источники данных

- Класс InputSource
- Конструкторы
  - InputSource(InputStream) – из байтового потока
  - InputSource(Reader) – из символьного потока
  - InputSource(systemId) – по URL

Обработчик событий (1)

- Интерфейс ContentHandler
- Класс DefaultHandler
- Методы
  - setDocumentLocator(Locator locator) – установить источник местоположения
  - startDocument() – начало документа
  - endDocument() – конец документа
  - startElement(ns, localName, qName, Attributes) – открывающий тег элемента
  - endElement(ns, localName, qName) – закрывающий тег элемента

Обработчик событий (2)

- Методы
  - characters(char[] ch, offset, len) – последовательность символов
  - ignorableWhitespace(char[] ch, offset, len) – последовательность пробельных символов
  - processingInstruction(prefix, data) – рекомендация по обработке
  - startPrefixMapping(prefix, uri) – начало области использование префикса
  - endPrefixMapping(prefix, uri) – окончание области использование префикса

Атрибуты

- Интерфейс Attributes
- Методы
  - getLength() – количество атрибутов
  - getLocalName(index) – локальное имя
  - getQName(index) – полное имя
  - getURI(index) – пространство имен
  - getValue(index) – получить значение по индексу
  - getValue(qName) – получить значение по полному имени
  - getValue(ns, localName) – получить значение по пространству имен и локальному имени

Информация о местоположении

- Интерфейс Locator
- Методы
  - getLineNumber() – номер строки
  - getColumnNumber() – номер столбца
  - getSystemId() – URL разбираемого файла

Обработка ошибок

- Интерфейс ErrorHandler
- Методы
  - error(SAXParseException) – сообщение об исправимой ошибке
  - fatalError(SAXParseException) – сообщение о неисправимой ошибке
  - warning(SAXParseException) – сообщение о предупреждении

Исключения

- Класс SAXException
- Методы
  - getLineNumber() – номер строки
  - getColumnNumber() – номер столбца
  - getSystemId() – URL разбираемого файла

Создание SAXParser

- Класс SAXParserFactory
- Методы
  - static newInstance() – создать фабрику
  - newSAXParser() – создать разборщик
  - setFeature(uri, value) – установить настройку
  - setNamespaceAware(value) – установить поддержку пространств имен
- Класс SAXParser implements XMLReader

Часть 4

DOM

Document Object Model

- Представляет XML-документ в виде дерева узлов
- Пакеты
  - org.w3c.dom – модель DOM
  - java.xml.parsers – разборщики

Узел (1)

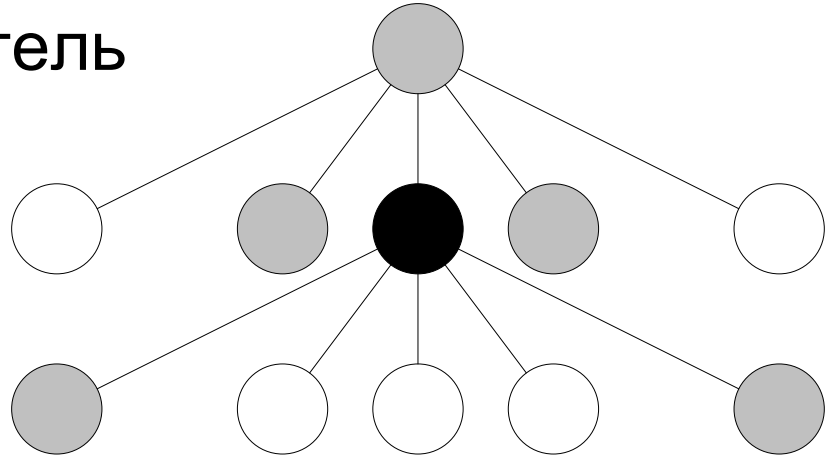
- Интерфейс Node
- Структура узла
  - getLocalName() – локальное имя
  - getNamespaceURI() – пространство имен
  - getPrefix() – префикс
  - getNodeName() – имя узла
  - getNodeValue() – значение узла
  - getNodeType() – тип узла

Типы узлов

Интерфейс	Описание	Имя	Значение
Attr	Атрибут	Имя	Значение
CDATASection	Блок символов	#cdata-section	Содержание
Comment	Комментарий	#comment	Содержание
Document	Документ	#document	
Element	Элемент	Имя	
ProcessingInstruction	Рекомендация по обработке	Имя	Значение
Text	Текст	#text	Содержание

Навигация по узлам

- Интерфейс Node
- Структура узла
  - getNextSibling() – предыдущий брат
  - getPreviousSibling() – следующий брат
  - getFirstChild() – первый ребенок
  - getLastChild() – последний ребенок
  - getParentNode() – родитель



Атрибуты

- Интерфейс NamedNodeMap
- Методы
  - getLength() – количество элементов
  - item(index) – элемент по индексу
  - getNamedItem(name) – элемент по имени
  - getNamesItemNS(namespace, localName) – элемент по имени и пространству имен
- Методы интерфейса Node
  - getAttributes() – получить атрибуты
  - hasAttributes() – проверить наличие атрибутов

Вложенные узлы

- Интерфейс NodeList
- Методы
  - getLength() – количество элементов
  - item(index) – элемент по индексу
- Методы интерфейса Node
  - getChildNodes() – получить детей
  - hasChildNodes() – проверить наличие детей

Элементы

- Интерфейс Element
- Методы
  - Работа с атрибутами
  - Работа с вложенными элементами

Разбор XML в DOM

- Класс DocumentBuilder
- Методы
  - parse(File | InputStream | InputSource | URI) – построить документ
  - isNamespaceAware() – поддерживает ли пространства имен

Создание DocumentBuilder

- Класс DocumentBuilderFactory
- Методы
  - static newInstance() – создать фабрику
  - newDocumentBuilder() – создать DocumentBuilder
  - setFeature(uri, value) – установить настройку
  - setNamespaceAware(value) – установить поддержку пространств имен
  - setIgnoringComments(value) – установить игнорирование комментариев
  - setIgnoringElementContentWhitespace(value) – пропуск текстовых узлов из одних пробелов

Java Advanced

Описание и проверка структуры XML

Часть 1

DTD

Document Type Definition

- DTD – язык описания структуры XML-документов
- Описание не является XML-документом
- DTD не предназначен для описания документов с пространствами имен

Ссылки на файлы

- Синтаксис
  - СсылкаНаФайл

    СсылкаНаСистемныйФайл |

    СсылкаНаОбщедоступныйФайл
  - СсылкаНаСистемныйФайл

    SYSTEM "ИмяФайла**"**
  - СсылкаНаОбщеизвесныйФайл

    PUBLIC "Идентификатор**" "**ИмяФайла**"**
- Примеры
  - SYSTEM "books.xml"
  - PUBLIC "-//Examples/BookExample" "books.xml"

Указание DTD

- Синтаксис
  - **<!DOCTYPE** ИмяКорневогоЭлемента ССылкаНаФайл? ВнутреннееОписание?**>**
  - ВнутреннееОписание ::= **[** ТелоDTD **]**
- Примеры
  - <!DOCTYPE books SYSTEM "books.dtd">
  - <!DOCTYPE books PUBLIC "-//Examples/BookDTD" "books.dtd">
  - <!DOCTYPE books […]>
  - <!DOCTYPE books SYSTEM "books.dtd" […]>

Описание элемента

- Описание структуры содержимого
  - Описываются возможные вложенные элементы и текст
- Описание атрибутов
  - Описываются имена, типы и значения атрибутов по умолчанию

Описание структуры содержимого

- Синтаксис
  - **<!ELEMENT** ИмяЭлемента Содержимое**>**
  - **Содержимое**
    - **EMPTY** **- без содержимого**
    - **ANY** **- любое содержимое**
    - **Дети** **- только вложенные элементы**
    - **Смешанное - вложенные элементы и текст**
- **Примеры**
  - **<!ELEMENT author EMPTY>**
  - **<!ELEMENT text ANY>**

Описание детей

- Синтаксис
  - (Выбор | Последовательность) Количество
  - Дети
    - (Имя | Выбор | Последовательность) Количество
  - Выбор
    - **(** Дети **|** Дети **|** …**|** Дети **)** Количество
  - Последовательность
    - **(** Дети **,** Дети **,** …**,** Дети **)** Количество
  - Количество
    - - Один
    - **?** - Ноль или один
    - **+** - Один и более
    - ***** - Любое

Примеры описания детей

- Только книги

  <! ELEMENT library (book)*>
- Книги и журналы вперемешку

  <! ELEMENT library (book | magazine)*>
- Сначала книги, потом журналы

  <! ELEMENT library (book*, magazine*)>
- Книги и журналы парами

  <! ELEMENT library (book, magazine)*>
- Блок кода

  <! ELEMENT body (begin?, (if | while | for)*, end?)>

Описание смешанного содержимого

- Синтаксис
  - **(#PCDATA |** Имя **|** Имя **|** … Имя **) ***
- Примеры
  - (#PCDATA | br | emboss | img)*
  - (#PCDATA)*
  - (#PCDATA)

Описание атрибутов

- Синтаксис
  - **<!ATTLIST** Имя ОписаниеАтрибута***>**
  - **Описание атрибута**
    - **Имя ТипАтрибута ЗначениеПоУмолчанию**

Типы атрибутов

- Строковые
  - **CDATA**
- Проверяемые
  - **ID** - **Идентификатор**
  - **IDREF** - **Ссылка на идентификатор**
  - **IDREFS** - **Ссылки на идентификатор**
  - **NMTOKEN** - **Имя**
  - **NMTOKENS** - **Имена**
- Перечислимые
  - **(Имя | Имя | … | Имя)**

Значения по умолчанию

- Значение по умолчанию
  - **"**значение**"**
- Значение по умолчанию
  - **#FIXED "**значение**"**
- Без значения по умолчанию
  - **#IMPLIED**
- Обязательно указывать
  - **#REQUIRED**

Пример описания атрибутов

- Рамка

```
<!ATTLIST border
    ID              ID                  #REQUIRED
    type            (single|double)     "single"
    color           CDATA               #REQUIRED
    width           CDATA               #IMPLIED
    direction       (ltor)              #FIXED "ltor"
>
```

Сущности

- Общие сущности
  - **<!ENTITY** Имя **"**значение**">**
  - **<!ENTITY** Имя СсылкаНаФайл**>**
- Сущности-параметры
  - **<!ENTITY %** Имя **"**значение**">**
  - **<!ENTITY %** Имя СсылкаНаФайл**>**
- Примеры
  - <!ENTITY file SYSTEM "books.xml">
  - <!ENTITY % statements "(if|while|for)">

Применение общих сущностей

- Сокращения

```
<!ENTITY copyright “© Georgiy Korneev 2005">

<p>Copyright: &copyright;</p>
```

- Разбиение XML на файлы

```
<!ENTITY books SYSTEM "books.xml">
<!ENTITY magazines SYSTEM "magazines.xml">

<library>
    &books; &magazines;
</library>
```

Применение сущностей-параметров

- Структура программы

```
<!ENTITY % statements "if | while | for">
<!ELEMENT while (%statements;)*>
<!ELEMENT for (%statements;)*>
<!ELEMENT if (then, else?)*>
<!ELEMENT then (%statements;)*>
<!ELEMENT else (%statements;)*>
```

- Разбиение DTD на файлы

```
<!ENTITY % books SYSTEM "books.dtd">
%books;
```

DTD и пространства имен

- В DTD можно указывать название элементов и атрибутов с двоеточиями
- Пространства имен задаются с помощью fixed-атрибутов

Пример: DTD и пространства имен

- Библиотека

```
<!ELEMENT lib:book (lib:author)*>

<!ATTLIST lib:book
    xmlns:lib          CDATA      #FIXED
                  "http://www.example.com/library“
    caption            CDATA      #REQUIRED
    isbn               CDATA      #IMPLIED
>
```

Часть 2

XML Schema

XML Schema

- XML Schema – язык описания структуры XML-документов
- Описание является XML-документом
- XML Schema предназначена для описания документов с пространствами имен
- XML Schema позволяет выразить все то же, что и DTD
- Пространство имен http://www.w3.org/2001/XMLSchema

Типы

- Описывают множество значений
- Простые типы (simpleType)
  - Строка символов
  - Атрибуты, Элементы
- Сложные типы (complexType)
  - Элементы и текст
  - Элементы

Простые типы

- Стандартные
- Ограниченные (restricted)
  - Ограничение другого типа
- Списочные (list)
  - Список элементов другого типа
- Объединение (union)
  - Объединение простых типов

Стандартные типы (1)

- Строковые
  - string, normalizedString, token
- Неограниченные целые
  - integer, positiveInteger, nonPositiveInteger, negativeInteger, nonNegativeInteger
- Ограниченные целые
  - (unsigned)Byte, (unsigned)Short, (usigned)Int, (unsigned)Long
- Дробные
  - decimal, float, double

Стандартные типы (2)

- Дата и время
  - dateTime, date, time, duration
- Имена
  - name, QName, NCName
- Из DTD
  - ID, IDREF, NMTOKEN
- Специальные
  - boolean, anyURI, language

Объявление ограниченного типа

- Синтаксис

```
<xsd:simpleType name="Имя">
    <xsd:restriction base="БазовыйТип">
        Ограничения
    </xsd:restriction>
</xsd:simpleType>
```

Пример: объявление типа

- Серийный номер

```
<xsd:simpleType name="serial">
    <xsd:restriction base="xsd:integer">
        <xsd:minInclusive value="10000"/>
        <xsd:maxInclusive value="99999"/>
    </xsd:restriction>
</xsd:simpleType>
```

- Телефон

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

Объявление списочного типа

- Синтаксис

```
<xsd:simpleType name="Имя">
    <xsd:list itemType="БазовыйТип">
</xsd:simpleType>
```

- Примеры

```
<xsd:simpleType name="serialList">
    <xsd:list itemType="serial"/>
</xsd:simpleType>
<xsd:simpleType name="phoneList">
    <xsd:list itemType="phone"/>
</xsd:simpleType>
```

Объявление типа-объединения

- Синтаксис

```
<xsd:simpleType name="ИмяТипа">
    <xsd:union memberTypes="список типов"/>
</xsd:simpleType>
```

- Пример

```
<xsd:simpleType name="phoneSerial">
    <xsd:union memberTypes="serial phone"/>
</xsd:simpleType>
```

Объявление сложного типа

- Синтаксис

```
<xsd:complexType
    name   = "ИмяТипа "
    mixed  = "boolean"
>
    ОписаниеСодержимого
    ОписаниеАтрибута*
</xsd:complexType>
```

Описание атрибута

- Синтаксис

```
<xsd:attribute
    default   = "ЗначениеПоУмолчанию"
    fixed     = "ЗначениеПоУмолчанию"
    name      = "Имя"
    ref       = "ИмяАтрибута"
    type      = "ИмяТипа"
    use       = "optional | prohibited | required"
>ОбявлениеПростогоТипа</xsd:attribute>
```

- Пример

```
<xsd:attribute name="serial" type="serial"
    use="required"/>
```

Описание содержимого

- Синтаксис
  - Описание содержимого
    - **<xsd:choice>**Содержимое**</xsd:choice>**
    - **<xsd:sequence>**Содержимое**</xsd:sequence>**
    - **<xsd:any/>**
  - Содержимое
    - ОписаниеЭлемента
    - ОписаниеГруппы

Описание элемента

- Синтаксис

```
<element
    name          = "Имя"
    ref           = "ИмяЭлемента"
    type          = "ИмяТипа"
    minOccurs     = "число"
    maxOccurs     = "число | unbounded"
/>ОбъявлениеТипа</element>
```

- Пример

```
<xsd:element name="address" type="address"
    maxOccurs="unbounded"/>
```

Описание групп

- Синтаксис

```
<group
    name        = "Имя"
    ref         = "ИмяГруппы"
    maxOccurs       = "число | unbounded"
    minOccurs       = "число"
>ОписаниеСодержимого</group>
```

- Пример

```
<xsd:group name="contact"><xsd:choice>
    <xsd:element name="phone" type="phone"/>
    <xsd:element name="address" type="address"/>
</xsd:choice></xsd:group>
```

Пример: описание сложного типа

- Адрес

```
<xsd:complexType name="address">
  <xsd:sequence>
    <xsd:element name="city" type="xsd:string"/>
    <xsd:element name="street" type="xsd:string"/>
    <xsd:element name="building" type="xsd:string"/>
    <xsd:element name="office" type="xsd:integer"/>
  </xsd:sequence>
</xsd:complexType>
```

Описание структуры

- Синтаксис

```
<xsd:schema
    targetNamespace       = "NS"
    elementFormDefault    = "(un)qualified"
    attributeFormDefault  = "(un)qualified"
>
    Описание Типов, Атрибутов,  Элементов
</ xsd: schema>
```

Пример: пространства имен

```
<xsd:schema targetNamespace="ns"
    elementFormDefault="unqualified"
    attributeFormDefault="unqualified“
>
  <xsd:element name="contact" type="ns:contact"/>
  <xsd:complexType name="contact"><xsd:choice>
      <xsd:element name="address" type="ns:address"/>
      <xsd:element name="phone" type="ns:phone"/>
  </xsd:choice></xsd:complexType>
  <xsd:simpleType name="phone">
    <xsd:restriction base="xsd:string">
    <xsd:pattern value="\d{3}-\d{2}-\d{2}"/>
    </xsd:restriction>
  </xsd:simpleType>
  …
</xsd:schema>
```

Пример: пространства имен

- elementFormDefault="unqualified"

```
<contact>
    <address>St. Petersburg, ...</address>
</contact>
```

- elementFormDefault="qualified"

```
<ns:contact
    xmlns:ns="ns"
>
    <ns:phone>123-45-67</ns:phone>
</ns:contact>
```

Разбиение на файлы

- Возможно включения одних схем в другие
- Синтаксис

```
<xsd:include
    namespace           = "URI"
    schemaLocation      = "URL"
/>
```

- Пример

```
<xsd:shema targetNamespace="namespace">
    <xsd:include
        namespace           ="contact"
        chemaLocation       ="file:/contact.xsd"
    />
</xsd:shema>
```

Аннотации

- Определение типов, элементов, атрибутов могут содержать аннотации
- Синтаксис
  - **<annotation>**Описания**</annotation>**
  - Описания
    - **<document>**текст**</document>**
    - **<appinfo>**текст**</appinfo>**

Связывание документа со схемой

- Пример

```
<contact
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-
    instance"
  xsi:schemaLocation="contact.xsd"
>
    ...
</contact>
```

Часть 3

Проверка структуры XML

Проверка по DTD

- Свойство DocumentBuilderFactory
  - validating – производить ли проверку
- Метод DocumentBuilder
  - isValidating() – производиться ли проверка
- Свойство SAXParserFactory
  - validating – производить ли проверку
- Свойство SAXParser
  - isValidating() – производиться ли проверка

Проверка по XML Schema

- Пакет javax.xml.validation
- Класс Shema
- Установка схемы
  - Свойство DocumentBuilderFactory.schema
  - Свойство SAXParserFactory.schema

Создание Shema

- Класс SchemaFactory
- Методы
  - static newInstance(type) – создание фабрики
  - newSchema(…) – создание экземпляра схемы
  - setFeature(uri, value) – установить настройку
- Типы схем
  - Класс XMLConstants
    - W3C_XML_SCHEMA_NS_URI
    - http://www.w3.org/2001/XMLSchema

Пример: задание схемы

```
SchemaFactory sfactory =
    SchemaFactory.newInstance(
        XMLConstants.W3C_XML_SCHEMA_NS_URI);
Schema schema = sfactory.newSchema(
        new File("books.xsd"));

SAXParserFactory factory =
        SAXParserFactory.newInstance();
factory.setSchema(schema);
factory.setNamespaceAware(true);
factory.setValidating(true);
SAXParser parser = factory.newSAXParser();
```