

# **XML Json**



# XML & JAVA

## ► Корректность XML

- синтаксическая корректность (well-formed), - синтаксические правила XML;
- действительность (valid) - данные соответствуют некоторому набору правил, определенных пользователем
  - DTD
  - XML-схемы (XSD)

# язык описания DTD (Document Type Definition)

## DTD отдельно

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>  
<! DOCTYPE students SYSTEM "students.dtd">
```

## DTD встроен в XML

```
<?xml version="1.0" ?>  
<! DOCTYPE student [  
<!ELEMENT student (name, telephone, address)>  
  
]>
```

# Схема XSD

```
<element name="telephone" type="positiveInteger" />
```

► schema – корневой элемент

► element

- name — имя элемента;
- type — тип элемента;
- ref — ссылается на определение элемента, находящегося в другом месте;
- minOccurs и maxOccurs — количество повторений (unbounded, по умолчанию 1)

```
<element name="card"  
         type="tns:Card"  
         minOccurs="1"  
         maxOccurs="unbounded" />
```

# Типы

## ► Простые

- не имеют атрибутов и дочерних элементов

## ► Сложные

- Содержат другие элементы

# Простой тип: правила

- ▶ `simpleType` - создает тип
- ▶ `name` - содержит имя типа
- ▶ Объявление - локально внутри элемента, глобально с использованием атрибута `name` для ссылки на тип
- ▶ `base` - указывает основной тип
- ▶ `restriction` :
  - `minInclusive` — минимальное число;
  - `maxInclusive` — максимальное значение типа;
  - `length` — длина значения;
  - `pattern` — шаблон значения, регулярное выражением;
  - `enumeration` — перечисление.

```
<simpleType name="Login">  
  <restriction base="ID">  
    <pattern value="(\\w){8, 20}" />  
  </restriction>  
</simpleType>
```

# Сложные типы

- ▶ `complexType`
- ▶ `name` - имя типа
- ▶ `sequence`, `all`, `choice` – задает опред. последовательность
- ▶ `element`
- ▶ `mixed = true` - м.с. текст
- ▶ `attribute` - атрибуты
  - `name` — имя атрибута
  - `type` — тип значения атрибута
  - `use` - `required`, `optional`, `prohibited`.
- ▶ `default` – знач. по умолч.
- ▶ `fixed` – фиксир. знач.



```
<complexType name="Student">
  <sequence>
    <element name="name" type="string"/>
    <element name="telephone" type="positiveInteger"/>
    <element name="address" type="tns:Address"/>
  </sequence>
  <attribute name="login" type="tns:Login" use="required"/>
  <attribute name="faculty" type="string" use="optional"/>
</complexType>
```

```
<attribute name="air">
  <simpleType>
    <restriction base="string">
      <enumeration value="Boing"></enumeration>
      <enumeration value="A300"></enumeration>
      <enumeration value="TY134"></enumeration>
    </restriction>
  </simpleType>
</attribute>
```

```

<?xml version="1.0" encoding="UTF-8" ?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
        xmlns:tns="http://www.bstu.by/bankaccount"
        targetNamespace="http://www.bstu.by/bankaccount" elementFormDefault="qualified">
  <element name="clients">
    <complexType>
      <sequence>
        <element name="client"
                  type="tns:client"
                  minOccurs="1"
                  maxOccurs="unbounded"/>

        </sequence>
      </complexType>
    </element>
    <complexType name="client">
      <sequence>
        <element name="firstname" type="string" minOccurs="1" maxOccurs="1"/>
        <element name="lastname" type="string" minOccurs="1" maxOccurs="1"/>
        <element name="personalaccount" type="tns:personalaccount" minOccurs="0"
                                                         maxOccurs="unbounded"/>
        <element name="depositaccount" type="tns:depositaccount" minOccurs="0"
                                                         maxOccurs="unbounded"/>
        <element name="creditaccount" type="tns:creditaccount" minOccurs="0"
                                                         maxOccurs="unbounded"/>

      </sequence>
      <attribute name="id" type="ID"/>
    </complexType>
  
```

```
<complexType name="personalaccount">
  <sequence>
    <element name="isblocked" type="boolean" minOccurs="1"
              maxOccurs="1"/>
    <element name="updatedate" type="date" minOccurs="1"
              maxOccurs="1"/>
    <element name="balance" type="double" minOccurs="1"
              maxOccurs="1"/>
  </sequence>
  <attribute name="id" type="ID"/>
</complexType>
<complexType name="creditaccount">
  <complexContent>
    <extension base="tns:personalaccount">
      <sequence>
        <element name="lendingrate" type="double" minOccurs="1" maxOccurs="1"/>
        <element name="month" type="int" minOccurs="1" maxOccurs="1"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

```
<simpleType name="Login">
  <restriction base="ID">
    <pattern value="([a-zA-Z])[a-zA-Z0-9]{7,19}" />
  </restriction>
</simpleType>
```

```
<complexType name="AirPassenger">
  <complexContent>
    <extension base="tns:air">
      <attribute name="count-passangers" type="positiveInteger"
        use="required" />
      <attribute name="model" default="boing">
        <simpleType>
          <restriction base="string">
            <enumeration value="boing"></enumeration>
            <enumeration value="TY"></enumeration>
            <enumeration value="IL"></enumeration>
            <enumeration value="Airbus"></enumeration>
          </restriction>
        </simpleType>
      </attribute>
    </extension>
  </complexContent>
</complexType>
```

```
public class ValidatorSAX {
    private static final Logger LOG = Logger.getLogger(ValidatorSAX.class);
    public void valid() {
        String filename = "files/airlist.xml";
        String schemaname = "files/air.xsd";
        String logname = "log/log.xml";
        Schema schems = null;

        String language = XMLConstants.W3C_XML_SCHEMA_NS_URI;
        SchemaFactory factory = SchemaFactory.newInstance(language);
        try{
            schems = factory.newSchema(new File(schemaname));
            Validator validator = schems.newValidator();
            Source source = new StreamSource(filename);

            AirHandler sh = new AirHandler(logname);
            validator.setErrorHandler(sh);
            validator.validate(source);
            LOG.info(filename + "is valid");
        }
        catch (SAXException e){
            LOG.error(filename + " SAX error " + e.getMessage());
        }
        catch (IOException e){
            LOG.error(" io error " + e.getMessage());
        }
    }
}
```

```
ValidatorSAX airvalid = new ValidatorSAX();  
airvalid.valid();
```

```
<airs>  
  <air-passenger uid = "35" >  
    <highth> fsfsfs </highth>  
    <length>sdadafsasf</length>  
    <weith> 37000</weith>  
    <rang> 12500</rang>  
    <max-speed>950</max-speed>  
    <count-passangers> 436</count-passangers>  
    <model> boing </model>  
  </air-passenger>
```

# JAХВ. Маршализация и демаршализация

- ▶ Маршализация — механизм преобразования данных из java-объектов в конкретное хранилище
- ▶ Демаршализация — обратный процесс преобразования данных из внешних источников в структуру хранения, поддерживаемую виртуальной машиной

```
@XmlRootElement
@XmlAccessorType(XmlAccessType.FIELD)
@XmlType(name = "CardEx", propOrder = {
    "login",
    "number",
}) // задание последовательности элементов XML

public class CardEx {
    @XmlAttribute(required = true)
    @XmlJavaTypeAdapter(CollapsedStringAdapter.class)
    @XmlID
    private String login;
    @XmlElement(required = true)
    private String number;

    public CardEx(String login, String number) {
        this.login = login;
        this.number = number;
    }

    public CardEx() {
    }
}
```



```
@XmlRootElement
```

```
class Cards {
```

```
    @XmlElement(name="card")
```

```
    private ArrayList<CardEx> list = new ArrayList<CardEx>()
```

```
    public Cards() {
```

```
        super();
```

```
    }
```

```
    public void setList(ArrayList<CardEx> list) {
```

```
        this.list = list;
```

```
    }
```

```
    public boolean add(CardEx st) {
```

```
        return list.add(st);
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Students [list=" + list + "];
```

```
    }
```

```
}
```

```

class MarshalMain {
    public static void main(String[] args) {
        try {
            JAXBContext context = JAXBContext.newInstance(Cards.class);
            Marshaller m = context.createMarshaller();
            Cards st = new Cards() { // анонимный класс
                { CardEx s = new CardEx("sssss", "qqqq");
                  this.add(s); }
            };
            m.marshal(st, new FileOutputStream("data/card marsh.xml"));
            m.marshal(st, System.out); // копия на консоль

        } catch (JAXBException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }
    }
}

```

```

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<cards><card login="sssss"><number>qqqq</number></card></cards>

```

CardHandler.java x SaxRunner.java x

```
package by.belstu.fit.isit.patsei;

/**
 * Created by Katarina Danil on 16.05.2017.
 */

public class CardHandler {

    protected int login;
    private String name;
    protected int sum;
    private String number;

    public CardHandler() {}
    public CardHandler(int login, String name, int sum, String number) {}

    public void setLogin(int login) {this.login = login;}
    public void setName(String name) {this.name = name;}
    public void setSum(int sum) {this.sum = sum;}
    public void setNumber(String number) {this.number = number;}

    public int getLogin() {return login;}
    public String getName() {return name;}
    public int getSum() {return sum;}
    public String getNumber() {return number;}

    @Override
    public String toString() {
        return "CardHandler{" +
            "login=" + login + ", " +
            "name=" + name + ", " +
            "sum=" + sum + ", " +
            "number=" + number + "
        ";
    }
}
```

Class name:

CardMMM

☒ Include parameter and return types of following methods:

Add to JAXB Generation	Parameter / return types of f...
<input checked="" type="checkbox"/>	void setNumber
<input checked="" type="checkbox"/>	void setName
<input checked="" type="checkbox"/>	void setSum
<input checked="" type="checkbox"/>	void setLogin
<input checked="" type="checkbox"/>	int getNumber
<input checked="" type="checkbox"/>	String getName
<input checked="" type="checkbox"/>	String getLogin
<input checked="" type="checkbox"/>	int getSum
<input checked="" type="checkbox"/>	String toString

OK

Cancel

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<xs:schema version="1.0"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
a">
```

```
<xs:complexType name="cardMMM">
  <xs:sequence>
    <xs:element name="login"
type="xs:string" minOccurs="0"/>
    <xs:element name="name"
type="xs:string" minOccurs="0"/>
    <xs:element name="number"
type="xs:int"/>
    <xs:element name="sum"
type="xs:int"/>
  </xs:sequence>
</xs:complexType>
</xs:schema>
```

WebServices

Generate Wsdl From Java Code...

Generate Xml Schema from Java using JAXB...

## ► JAXB. Генерация классов

- инжиниринг классов на языке Java на основе XML-схемы (XSD-схемы)

# JAXP Java API for XML Processing

## ► DOM (Document Object Model)

- строит древовидную модель
- Узлы, связанные отношением род.

платформенно-независимый API, позволяющий программам и скриптам управлять содержимым документов HTML и XML

## ► SAX (Simple API for XML) -

при чтении/анализе документа, анализатор вызывает методы, связанные с различными участками XML-файла

- модель последовательной одноразовой обработки (не создает внутренних деревьев)
- При прохождении по XML генерируют квазисобытия об обнаружении эл

## ► StAX (Streaming API for XML)

- не создает дерево объектов в памяти
- ждут команды от приложения для перехода к следующему элементу XML - итератор
- API для создания XML-документа

# Псевдособытийная модель SAX–анализаторы

► Интерфейс `org.xml.sax.ContentHandler`

```
void startDocument ();
```

```
void endDocument ();
```

```
void startElement  
    (String uri,  
     String localName,  
     String qName,  
     Attributes attrs);
```

```
void endElement  
    (String uri, String localName, String qName);
```

```
void characters (char[] ch, int start, int length)
```

# SAX2 API

org.xml.sax

интерфейсы

DTDHandler

DocumentHandler

EntityResolver

# Алгоритм обработки

1. Создать класс, реализующий интерфейсы (ContentHandler, ErrorHandler, DTDHandler, EntityResolver, DocumentHandler)  
+ реализовать методы
2. Создать `org.xml.sax.XMLReader`
3. Передать в `XMLReader` объект класса из п.1: `setContentHandler()`, `setErrorHandler()`, `setDTDHandler()`, `setEntityResolver()`
4. Вызвать `parse(String filename)` класса `XMLReader`



# Пример

```
<?xml version="1.0" encoding="UTF-8" ?>
<cards xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://www.patsei.by">

  <card id="c1">
    <name>Igor</name>
    <number>34</number>
    <sum>13.50</sum>
  </card>
  <card id="c3">
    <name>Diman</name>
    <number>123</number>
    <sum>200007</sum>
  </card>
</cards>
```

# 1. Класс парсера

```
public class DemoCardHandler extends DefaultHandler {  
    //реагирует на событие начала документа  
    @Override  
    public void startDocument() throws SAXException {  
        System.out.println("Doc started");  
    }  
    @Override  
    public void endDocument() throws SAXException {  
        System.out.println("\nDoc ended");  
    }  
    @Override  
    public void startElement  
(String uri, String localName, String qname, Attributes attrs)  
    throws SAXException {  
        // "name".equals(localName);  
        String str = "";  
        // получение и вывод информации об атрибутах элемента  
        for (int i = 0; i < attrs.getLength(); i++) {  
            str += " " + attrs.getLocalName(i) + " = «  
                + attrs.getValue(i);  
        }  
        System.out.print(str.trim());    }
```

локальное имя элемента

комбинация

пространство имен

@Override

```
public void endElement  
    (String uri, String localName, String qName)  
        throws SAXException {  
    System.out.print("  " + localName);  
}
```

строку-значение

точка старта в строке

@Override

```
public void characters  
    (char[] chars, int start, int length)  
        throws SAXException {  
    System.out.print(new String(chars, start, length));  
}
```

}

# 2.+3.+4.

```
public class SaxMain {  
  
    public static void main(String[] args) {  
        try {  
            // создание SAX-анализатора из фабрики  
            XMLReader reader = XMLReaderFactory.createXMLReader();  
            DemoCardHandler handler = new DemoCardHandler();  
            // регистрируем  
            reader.setContentHandler(handler);  
  
            reader.parse("data/cards.xml"); //запускаем  
        }  
        catch (SAXException e) {  
            System.out.print("ошибка SAX парсера " + e);  
        }  
        catch (IOException e) {  
            System.out.print("ошибка I/O потока " + e);  
        }  
    }  
}
```

"C:\Program ...

Doc started

id = c1

Igor name

34 number

135340 sum

card

id = c3

Diman name

123 number

200007 sum

card

cards

Doc ended

Process finished with exit code 0

# Пример – запись в коллекцию объектов

1.

```
public class CardMMM {  
  
    protected int number;  
    private String name;  
    protected int sum;  
    private String login;  
  
    public CardMMM() {...}  
    public CardMMM(int number, String name, int sum) {...}  
  
    public void setNumber(int number) { this.number = number;  
    public void setName(String name) { this.name = name; }  
    public void setSum(int sum) { this.sum = sum; }  
    public void setLogin(String login) { this.login = login; }  
  
    public int getNumber() { return number; }  
    public String getName() { return name; }  
    public String getLogin() { return login; }  
    public int getSum() { return sum; }  
  
    @Override  
    public String toString() {...}  
}
```

```
<?xml version="1.0"  
encoding="UTF-8"?>  
<cards  
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  
xmlns="http://www.patsei.kh"  
  
    <card id="c1">  
        <name>Igor</name>  
        <number>34</number>  
        <sum>13.50</sum>  
    </card>  
  
    <card id="c3">  
        <name>Diman</name>  
        <number>123</number>  
        <sum>200007</sum>  
    </card>  
</cards>
```

```

public class CardHandler extends DefaultHandler {

    String thisElement = "";
    private Set<CardMMM> cards;
    private CardMMM current = null;

    public CardHandler() {
        cards = new HashSet<CardMMM>();
    }

    public Set<CardMMM> getCards() {
        return cards;
    }

    public void startElement
(String uri, String localName, String qName, Attributes
attrs) {
        if ("card".equals(localName)) {
            current = new CardMMM();
            current.setLogin(attrs.getValue(0));
        }
        thisElement = qName;
    }
}

```

2.

```
public void endElement
    (String uri, String localName, String qName) {
    if ("card".equals(localName)) {
        cards.add(current);
    }
    thisElement="";
}
```

```
public void characters(char[] ch, int start, int length) {
    String s = new String(ch, start, length).trim();
    if (thisElement.equals("name")) {
        current.setName(new String(ch, start, length));
    }
    if (thisElement.equals("sum")) {
        current.setSum(Integer.parseInt(s));
    }
    if (thisElement.equals("number")) {
        current.setNumber(Integer.parseInt(s));
    }
}
```



```
public class SaxRunner {
```

```
    public static void main(String[] args) {
```

3.

```
        try {
```

```
            CardHandler sh = new CardHandler();
```

```
            XMLReader reader = null;
```

```
            reader = XMLReaderFactory.createXMLReader();
```

```
            reader.setContentHandler(sh);
```

```
            reader.parse("data/cards.xml");
```

```
            Set<CardMMM> a = sh.getCards();
```

```
            System.out.print(sh.getCards());
```

```
        } catch (IOException e) {
```

```
            e.printStackTrace();
```

```
        } catch (SAXException e) {
```

```
            e.printStackTrace();
```

```
        }
```

```
    }
```

```
    "C:\Program ...
```

```
    [CardMMM{number=123, name='Diman', sum=200007, login='c3'}, CardMMM{number=34, name='Igor', sum=135340, login='c1'}]
```

```
    Process finished with exit code 0
```

# Древовидная модель DOM JAXP

Интерфейсы ПАКЕТА `org.w3c.dom`

имеют готовый набор методов для манипуляции деревом объектов и не зависят от конкретной реализации используемого анализатора

Стандартные DOM-анализаторы (JDK):

Xerces

**JAXP**

Сторонние библиотеки

# Интерфейсы JAXP

## ► org.w3c.dom.Document

- Доступ к корневому элементу

```
Element getDocumentElement()
```

## ► org.w3c.dom.Node

- Общий элемент дерева

<b>short</b>	getNodeTypes()
String	getNodeValue()
Node	getParentNode()
NodeList	getChildNodes()
NamedNodeMap	getAttributes()

## ► org.w3c.dom.Element

- для работы с элементом XML-документа (наследник Node)

String	getTagName (String name)
<b>boolean</b>	hasAttribute ()
String	getAttribute (String name)
Attr	getAttributeNode (String name)
NodeList	getElementsByTagName (String name)

## ► org.w3c.dom.Attr

- работы с атрибутами элемента

String	getName ()
Element	getOwnerElement ()
String	getValue ()
<b>boolean</b>	isId ()

# Пример

*разбор документа с использованием DOM-анализатора и инициализация на его основе множества объектов*

```
public class CardDOMParser {  
  
    private Set<CardMMM> cards;  
    private DocumentBuilder docBuilder;  
  
    public CardDOMParser() {  
        this.cards = new HashSet<CardMMM>();  
        // создание DOM-анализатора  
        DocumentBuilderFactory factory =  
            DocumentBuilderFactory.newInstance();  
        try {  
            docBuilder = factory.newDocumentBuilder();  
        } catch (ParserConfigurationException e) {  
            System.err.println  
                ("Ошибка конфигурации парсера: " + e);  
        }  
    }  
}
```

```
public Set<CardMMM> getCards() {  
    return cards;  
}
```

```
public void buildSetCards(String fileName)  
    throws IOException, SAXException {  
    Document doc = null;  
    // parsing XML-документа и создание  
    // древовидной структуры  
    doc = docBuilder.parse(fileName);  
    Element root = doc.getDocumentElement();  
    // получение списка дочерних элементов <card>  
    NodeList cardsList =  
        root.getElementsByTagName("card");  
    for (int i = 0; i < cardsList.getLength(); i++) {  
        Element cardElement = (Element) cardsList.item(i);  
        CardMMM current = buildCardMMM(cardElement);  
        cards.add(current);  
    }
```

```
private CardMMM buildCardMMM(Element cardElement) {  
  
    CardMMM tempCard = new CardMMM();  
    // заполнение объекта  
  
    tempCard.setLogin(cardElement.getAttribute("id"));  
    tempCard.setName(getElementTextContent(cardElement, "name"));  
    Integer number =  
        Integer.parseInt(getElementTextContent(  
            cardElement, "number"));  
    tempCard.setNumber(number);  
  
    Integer sum = Integer.parseInt(getElementTextContent(  
        cardElement, "sum"));  
    tempCard.setSum(sum);  
  
    return tempCard;  
}
```

*// получение текстового содержимого тега*

```
private static String getElementTextContent  
    (Element element, String elementName)  
{  
  
    NodeList nList =  
        element.getElementsByTagName(elementName);  
  
    Node node = nList.item(0);  
  
    String text = node.getTextContent();  
  
    return text;  
}
```



```
public class DomRunner {  
  
    public static void main(String[] args) {  
        try {  
            CardDOMParser domBuilder = new CardDOMParser();  
  
            domBuilder.buildSetCards("data/cards.xml");  
  
            System.out.println(domBuilder.getCards());  
            ...  
        }  
    }  
}
```

# StAX (Streaming API for XML) pull-парсером

► `javax.xml.stream`

► классы

- `XMLInputFactory`, `XMLStreamReader`
- `XMLOutputFactory`, `XMLStreamWriter`,

# Алгоритм

## 1) Получить ссылку XMLStreamReader

```
StringReader input = new StringReader("data/cards.xml");  
  
// InputStream input = new FileInputStream(new File("data/cards.xml"));  
  
XMLInputFactory inputFactory = XMLInputFactory.newInstance();  
XMLStreamReader reader = inputFactory.createXMLStreamReader(input);
```

## 2) Навигация

```
boolean hasNext()  
int next()
```

## 3) Методы

```
String getLocalName()  
String getAttributeValue(String namespaceURI, String localName)  
String getAttributeValue(int index)  
String getText()
```

# Пример

```
public class CardStaxParser {  
  
    private HashSet<CardMMM> cards = new HashSet<>();  
    private XMLInputFactory inputFactory;  
  
    public CardStaxParser() {  
        inputFactory = XMLInputFactory.newInstance();  
    }  
    public Set<CardMMM> getCrads() {  
        return cards;  
    }  
}
```

```

public void buildSetCards(String fileName) {
    FileInputStream inputStream = null;
    XMLStreamReader reader = null;
    String name;
    try {
        inputStream = new FileInputStream(new File(fileName));
        reader =
inputFactory.createXMLStreamReader(inputStream);
        // StAX parsing
        while (reader.hasNext()) {
            int type = reader.next();
            if (type == XMLStreamConstants.START_ELEMENT) {
                name = reader.getLocalName();
                if (name.equals("card")) {
                    CardMMM st = buildCard(reader);
                    cards.add(st);
                }
            }
        }
    } catch (XMLStreamException ex) ...
    } catch (FileNotFoundException ex) ...
    } finally {    inputStream.close();
        catch (IOException e)...
    }
}

```

```
private CardMMM buildCard (XMLStreamReader reader)
                                throws XMLStreamException {
    CardMMM st = new CardMMM();
    st.setLogin(reader.getAttributeValue(0));
    String name;
    int type;
    while (reader.hasNext()) {
        type = reader.next();
        switch (type) {
            case XMLStreamConstants.START_ELEMENT:
                name = reader.getLocalName();
                if (name.equals("name"))
                    st.setName(getXMLText(reader));
                if (name.equals("number"))
                    st.setNumber(Integer.parseInt(getXMLText(reader)));
                if (name.equals("sum"))
                    st.setSum(Integer.parseInt(getXMLText(reader)));
                break;
            case XMLStreamConstants.END_ELEMENT:
                name = reader.getLocalName();
                if (name.equals("card"))
                    return st;
        }
        throw new XMLStreamException("Unknown element in card");
    }
}
```

```

private String getXMLText(XMLStreamReader reader)
                                throws XMLStreamException {
    String text = null;
    if (reader.hasNext()) {
        reader.next();
        text = reader.getText();
    }
    return text;
}
}

```

```

public class StaxRunner {
    public static void main(String[] args) {

        CardStaxParser staxBuilder = new CardStaxParser();

        staxBuilder.buildSetCards("data/cards.xml");

        System.out.println(staxBuilder.getCards());
    }
}

```

StaxRunner StaxRunner StaxRunner StaxRunner

"C:\Program ...

[CardMM{number=123, name='Diman', sum=200007, login='c3'}, CardMM{number=34, name='Igor', sum=135340, login='c1'}]

Process finished with exit code 0

DOM-анализаторы	SAX/StAX-анализаторы
Нужно знать структуру документа	Нужно извлечь информацию о нескольких элементах
Понадобиться изменять структуру	
Информация из XML-документа будет использоваться несколько раз	Информация из документа нужна только один раз



# XSL

- ▶ стандарты стилевых таблиц W3C
  - CSS (Cascading Stylesheet)
  - XSL (XML Stylesheet Language)
    - ▶ XSLT (XSL Transformation)
    - ▶ XPath (язык путей и выражений)
    - ▶ XSLFO (XSL Formatting Objects).

# XSLT (XSL Transformation)

- удаление существующих или добавление новых элементов в XML-документ;
- создания нового XML-документа на основании заданного;
- извлечения информации из XML-документа с разной степенью детализации;
- преобразования XML-документа в документ HTML или текстовый документ другого типа.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:template match="/">
  </xsl:template>

</xsl:stylesheet>
```

- stylesheet - Корневой элемент таблицы
- output - Определяет формат результата
- choose - Выбор условия
- copy-of - создает копию
- element - создает элемент
- for-each - задает цикл обработки
- sort, text, value-of - извлечение элемента

`<xsl:template>` - задает шаблон преобразования

`match` - указание, к чему будет применяться

`<xsl:apply-templates>` - перейти к дочерним узлам

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Trans
```

```
    <xsl:output method="html"/>
```

```
  <xsl:template match="/cards/card">
```

```
    <xsl:for-each select="card">
```

```
      <xsl:value-of select="."/>
```

```
    </xsl:for-each>
```

```
  </xsl:template>
```

```
</xsl:stylesheet>
```

```
public class XSLTransformer {
    public static void main(String[] args) {
        try {
            TransformerFactory xstf = TransformerFactory.newInstance();
            // установка используемого XSL-преобразования
            Transformer transformer = null;

            transformer =
                xstf.newTransformer(new StreamSource("data/all.xml"));

            // установка исходного XML-документа и конечного XML-файла

            transformer.transform(new StreamSource("data/cards.xml"),
                                new StreamResult("data/info.html"));
        }
        catch (TransformerConfigurationException e) {
            e.printStackTrace();
        }
        catch (TransformerException e) {
            e.printStackTrace();
        }
    }
}
```

# JSON парсеры (сериализаторы)

## ► Data bind

- Java класс (аннотированный) → в иерархию json (можно наоборот)
- Аналог - JAXB Недостаток: скорость и память

## ► Tree Model

- json → Java классов – Node, JsonElement нужно обойти и получить из них информацию
- Аналог - DOM xml

## ► Streaming API

- низкоуровневый способ ручного разбора токенов json'a
- нет ограничений по памяти и хорошая производительность

## ► Аналог Xpath

- получает информацию из json'a по сложным критериям
- сложно получать всю информацию из json'a
- не формирует json



# Библиотеки

Способ	<u>Fastjson</u>	<u>Gson</u>	<u>Logan Square</u>	<u>JSON java</u>	<u>ig json parser</u>	<u>Jackson</u>	<u>Genson</u>	<u>JsonPath</u>
1. Data bind	<u>Да</u>	<u>Да</u>	<u>Да</u>	-	<u>Да</u>	<u>Да</u>	<u>Да</u>	-
2. Tree Model	-	<u>Да</u>	-	<u>Да</u>	-	<u>Да</u>	-	-
3. Streami ng API	-	<u>Да</u>	-	-	-	<u>Да</u>	-	-
4. Аналог и XPath	<u>Да</u>	-	-	-	-	-	-	<u>Да</u>

# Data bind

```
//-----Fastjson
//  to json
String jsonString1 = JSON.toJSONString(card);

//  from json
Card newCard = JSON.parseObject(jsonString1, Card.class);

//-----Gson
//  to json
Gson gson = new Gson();
String jsonString2 = gson.toJson(card);

//  from json
Card newCard2 = gson.fromJson(jsonString2, Card.class);
```

# Tree Model

Методы парсинга json'a:

Действие	<u>Gson</u>	<u>Jackson</u>
Инициализация	<code>JsonParser parser = new JsonParser()</code>	<code>new ObjectMapper()</code>
Парсинг json'a	<code>parser.parse(&lt;строка&gt;)</code>	<code>mapper.readValue(&lt;строка&gt;, JsonNode.class)</code>
Получение главного объекта	<code>root.getAsJsonObject()</code>	-
Получение строки	<code>root.get(&lt;имя&gt;).getString()</code>	<code>root.get(&lt;имя&gt;).asText()</code>
Получение дочернего объекта	<code>root.getAsJsonObject(&lt;имя&gt;)</code>	<code>root.get(&lt;имя&gt;)</code>

## Методы генерации json'a:

Действие	<u>Gson</u>	<u>Jackson</u>
Инициализация	-	<code>new ObjectMapper()</code>
Создание главного объекта	<code>new JsonObject()</code>	<code>mapper.createObjectNode()</code>
Добавить строковое поле	<code>root.addProperty(&lt;имя&gt;, &lt;строка&gt;)</code>	<code>root.put(&lt;имя&gt;, &lt;строка&gt;)</code>
Добавить дочерний объект	<code>root.add(&lt;имя&gt;, &lt;объект&gt;);</code>	<code>root.putObject(&lt;имя&gt;)</code>

*//Чтение Gson*

```
JsonParser parser = new JsonParser();  
JsonElement jsonElement = parser.parse(jsonStr);  
  
JsonObject rootObject = jsonElement.getAsJsonObject();  
                // чтение объекта  
String message = rootObject.get("id").getString();  
JsonObject childObject = rootObject.getAsJsonObject("card");  
String name = childObject.get("name").getString();
```

*// Генерация Gson*

```
JsonObject rootObject = new JsonObject();  
                // создаем главный объект  
rootObject.addProperty("id", "123");  
JsonObject childObject = new JsonObject();  
childObject.addProperty("name", "Dima");  
rootObject.add("card", childObject);  
  
Gson gson = new Gson();  
String json = gson.toJson(rootObject);  
                // генерация json строки
```

// Чтение Jackson

```

ObjectMapper mapper = new ObjectMapper();
JsonNode rootNode = mapper.readValue(jsonStr, Card.class);
// парсинг текста
String message = rootNode.get("id").asText();
JsonNode childNode = rootNode.get("card");
String name = childNode.get("name").asText();

```

// Генерация Jackson

```
OutputStream outputStream = new ByteArrayOutputStream();
ObjectMapper mapper = new ObjectMapper();
ObjectNode rootNode = mapper.createObjectNode();
// создание главного объекта
rootNode.put("id", "123");
ObjectNode childNode = rootNode.putObject("card");
childNode.put("name", "Dima");
mapper.writeValue(outputStream, childNode);
// запись json строки
```

# Streaming API

## Методы парсинга json'a:

Действие	<u>Gson</u>	<u>Jackson</u>
Инициализация	-	<code>new JsonFactory()</code>
Парсинг json'a	<code>reader = new JsonReader(&lt;input_stream&gt;)</code>	<code>parser = jsonFactory.createParser(&lt;строка&gt;)</code>
Проверка есть ли ещё токены	<code>reader.hasNext()</code>	<code>parser.hasCurrentToken()</code>
Получение типа токена	<code>reader.peek()</code>	<code>parser.nextToken()</code>
Получение следующего токена	<code>reader.nextString() reader.beginObject() reader.endObject() и т.п.</code>	<code>parser.nextToken()</code>
Пропуск токена	<code>reader.skipValue()</code>	<code>parser.nextToken()</code>
Получение строки	<code>reader.nextString()</code>	<code>parser.getText()</code>

## Методы генерации json'a:

Действие	<u>Gson</u>	<u>Jackson</u>
Инициализация	<code>writer = new JsonWriter(&lt;output_stream&gt;)</code>	<code>generator = new JsonFactory().createGenerator(&lt;output_stream&gt;, &lt;кодировка&gt;)</code>
Токен начала объекта	<code>writer.beginObject()</code>	<code>generator.writeStartObject()</code>
Токен окончания объекта	<code>writer.endObject()</code>	<code>generator.writeEndObject()</code>
Токен имени поля	<code>writer.name(&lt;имя&gt;)</code>	<code>generator.writeFieldName(&lt;имя&gt;)</code>
Токен строкового значения	<code>writer.value(&lt;строка&gt;)</code>	<code>generator.writeStringField(&lt;имя&gt;, &lt;строка&gt;)</code>



# Документация

- ▶ <https://github.com/google/gson/blob/master/UserGuide.md>
- ▶ <https://github.com/FasterXML/jackson-docs>

