XML

XML (англ. eXtensible Markup Language — расширяемый язык разметки). Рекомендован Консорциумом Всемирной паутины (W3C).

Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому).

Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

Любой XML документ должен иметь объявление, в котором указывается версия языка, кодировка, и корневой тэг.

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
</root>
```

Тэги могу вкладываться друг в друга образуя древовидную структуру, а так же иметь атрибуты.

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
    <child id="1">
        <grandchild id="1">
            Grand Child 1
        </grandchild>
    </child>
    <!-- Some meaningful comment -->
    <child id="2">
        <grandchild id="2">
            Grand Child 2
        </grandchild>
    </child>
</root>
```

Для чтения xml JDK предоставляет набор инструментов:

- DOM — читает весь документ в память;
- SAX — событийный парсер, на каждое событие вызывает пользовательский callback;
- StAX — потоковый парсер, пользователь управляет хэндлером, перемещая его по событиям;
- XPath — позволяет адресовать конкретные данные в документе.

Используя эти инструменты будем читать следующий документ:

```
<?xml version="1.0" encoding="utf-8" ?>
<catalog>
    <book id="bk101">
        <author>Gambardella, Matthew</author>
        <title>XML Developer's Guide</title>
        <genre>Computer</genre>
        <price>44.95</price>
        <publish_date>2000-10-01</publish_date>
        <description>An in-depth look at creating applications
            with XML.</description>
    </book>
    <book id="bk102">
        <author>Ralls, Kim</author>
        <title>Midnight Rain</title>
        <genre>Fantasy</genre>
        <price>5.95</price>
        <publish_date>2000-12-16</publish_date>
        <description>A former architect battles corporate zombies,
            an evil sorceress, and her own childhood to become queen
            of the world.</description>
    </book>
</catalog>
```

DOM parser

DOM parser позволяет создавать в памяти объектную модель XML документа, что дает возможность не только читать, но и изменять документ.

Парсинг происходит следующим образом:

```
// Инициализируем DOM parser
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();

// Разбираем документ
Document document = builder.parse(
            DOMParserExample.class.getClassLoader().getResourceAsStream("books.xml"));
```

После того как DOM построена, можно приступать к разбору элементов:

```
// Читаем корневой элемент
Element catalogEl = document.getDocumentElement();

// Читаем подузлы
NodeList books = catalogEl.getChildNodes();
```

```java
for (int i = 0; i < books.getLength(); i++) {
    // Выбираем подузел
    Node bookNode = books.item(i);

    // Проверяем что наш узел является элементом
    if (bookNode instanceof Element) {
        Book book = new Book();
        catalog.books.add(book);

        // Читаемм атрибуты
        NamedNodeMap attrs = bookNode.getAttributes();

        // Читаем id атрибут
        book.id = attrs.getNamedItem("id").getNodeValue();

        // Смотрим подузлы
        NodeList bookDetails = bookNode.getChildNodes();

        // Продолжение на следующем слайде
    }
}
```

```java
// Смотрим подузлы
NodeList bookDetails = bookNode.getChildNodes();

for (int j = 0; j < bookDetails.getLength(); j++) {
    Node node = bookDetails.item(j);

    if (node instanceof Element) {
        // Читаем каждый подузел
        String nodeName = node.getNodeName();
        String val = node.getTextContent();

        if ("author".equals(nodeName))
            book.author = val;
        else if ("title".equals(nodeName))
            book.title = val;
        else if ("genre".equals(nodeName))
            book.genre = val;
        else if ("price".equals(nodeName))
            book.price = Float.parseFloat(val);
        else if ("publish_date".equals(nodeName))
            book.publishDate = LocalDate.parse(val);
        else if ("description".equals(nodeName))
            book.description = val;
    }
}
```

Так же он позволяет разбирать документ при помощи XPath:

```
XPath xPath = XPathFactory.newInstance().newXPath();
NodeList nodeList = (NodeList) xPath.compile("/catalog/book")
                                    .evaluate(document, XPathConstants.NODESET);

for (int i = 0; i < nodeList.getLength(); i++) {
    Node node = nodeList.item(i);

    if (node.getNodeType() == Node.ELEMENT_NODE) {
        Book book = new Book();
        catalog.books.add(book);

        Element element = (Element) node;

        book.id = element.getAttribute("id");

        book.author = content(element, "author");
        book.title = content(element, "title");
        book.genre = content(element, "genre");
        book.price = Float.parseFloat(content(element, "price"));
        book.publishDate = LocalDate.parse(content(element, "publish_date"));
        book.description = content(element, "description");
    }
}


private static String content(Element element, String tagName) {
    return element.getElementsByTagName(tagName).item(0).getTextContent();
}
```

Создание документа происходит приблизительно тем же образом:

```
// Инициализируем фабрику и билдер
DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
DocumentBuilder builder = factory.newDocumentBuilder();

// Создаем документ и указываем версию
Document document = builder.newDocument();
document.setXmlVersion("1.0");

// Создаем корневой элемент
Element catalogEl = document.createElement("catalog");
document.appendChild(catalogEl);

// Добавляем подузлы
for (Book book : catalog.books) {
    Element bookEl = document.createElement("book");
    catalogEl.appendChild(bookEl);

    bookEl.setAttribute("id", book.id);

    appendDataNode(document, bookEl, "author", book.author);
    appendDataNode(document, bookEl, "title", book.title);
    appendDataNode(document, bookEl, "genre", book.genre);
    appendDataNode(document, bookEl, "price", String.valueOf(book.price));
    appendDataNode(document, bookEl, "publish_date", book.publishDate.toString());
    appendDataNode(document, bookEl, "description", book.description);
}

private static void appendDataNode(Document document, Element parent, String tagName,
                                                                    String value) {
    Element element = document.createElement(tagName);
    element.appendChild(document.createTextNode(value));
    parent.appendChild(element);
}
```

```
// Создаем трансформер
TransformerFactory transformerFactory = TransformerFactory.newInstance();
transformerFactory.setAttribute("indent-number", 4);

Transformer transformer = transformerFactory.newTransformer();
transformer.setOutputProperty(OutputKeys.INDENT, "yes");

// Оборачиваем докумен
DOMSource src = new DOMSource(document);
StreamResult res = new StreamResult(new FileOutputStream("books.xml"));

// Создаем конечный XML
transformer.transform(src, res);
```

books.xml:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<catalog>
   <book id="001">
      <author>IFMO community</author>
      <title>How to parse XML in Java</title>
      <genre>Erotic</genre>
      <price>0.0</price>
      <publish_date>2017-01-08</publish_date>
      <description>The definitive guide.</description>
   </book>
</catalog>
```

SAX parser

В случае когда имеем дело с большими документами, то загрузка всего XML в память может оказаться весьма ресурсоемкой операцией. Событийные/потоковые парсеры позволяют обрабатывать его «по ходу дела», минимально нагружая память.

Одним из таких парсеров является SAX.

Суть работы с ним заключается в том, что проходя документ на каждое событие (начало документа, открывающий, закрывающий тэги, текст и т. д.) он вызывает пользовательский колбэк.
Иными словами пользователь пассивно слушает о присходящих событиях.

```java
// Инициализируем SAX фабрику
SAXParserFactory factory = SAXParserFactory.newInstance();

InputStream in = SaxExample.class.getClassLoader().getResourceAsStream("books.xml");

// Создаем SAX парсер
SAXParser saxParser = factory.newSAXParser();

// Инициализируем наш хэндлер
BookHandler handler = new BookHandler();

// Стартуем парс
saxParser.parse(in, handler);

in.close();

// Читаем получившийся результат
System.out.println(handler.catalog);
```

```
private static class BookHandler extends DefaultHandler {
    private Catalog catalog = new Catalog();
    private Book book;
    private StringBuilder value = new StringBuilder();
    @Override
    public void startElement(String uri, String localName, String qName, Attributes attributes)
                                                                                throws SAXException {
        // Метод вызывается когда, встречается открывающий тэг
        if ("book".equals(qName)) {
            book = new Book();
            // Читаем атрибут
            book.id = attributes.getValue("id");
            catalog.books.add(book);
        }
    }
    @Override
    public void endElement(String uri, String localName, String qName) throws SAXException {
        // Метод вызывается, когда встречается закрывающий тэг
        String val = value.toString().trim();

        if ("author".equals(qName))
            book.author = val;
        else if ("title".equals(qName))
            book.title = val;
        else if ("genre".equals(qName))
            book.genre = val;
        else if ("price".equals(qName))
            book.price = Float.parseFloat(val);
        else if ("publish_date".equals(qName))
            book.publishDate = LocalDate.parse(val);
        else if ("description".equals(qName))
            book.description = val;

        value.setLength(0);
    }
    @Override
    public void characters(char[] ch, int start, int length) throws SAXException {
        // Метод вызывается, когда встречается текст
        value.append(ch, start, length);
    }
}
```

StAX parser

StAX парсер так же как и SAX потоковый парсер, за тем исключением, что пользователь не пассивный слушатель событий, а управляет перемещением курсора самостоятельно.

В определенных случаях такой подход дает дополнительную гибкость.

API StAX парсера делится на две категории:

- Iterator API — позволяет работать как с итератором, т. е. получать следующее событие, которое можно сохранить и использовать позже;
- Cursor API — не создает лишних объектов и дает максимальную производительность.

В отличие от SAX StAX умеет создавать XML.

Инициализация происходит как и двух предыдущих парсеров:

```
// Инициализируем фабрику и Reader
XMLInputFactory factory = XMLInputFactory.newFactory();

InputStream in = SaxExample.class.getClassLoader().getResourceAsStream("books.xml");

XMLEventReader reader = factory.createXMLEventReader(in);
```

После чего итерируемся по всем (или только по нужным) элементам:

```
// Итерируемся по всем элементам
while (reader.hasNext()) {
    XMLEvent event = reader.nextEvent();

    // Обработчики
}
```

```
// Обрабатываем открывающий тэг
if (event.getEventType() == XMLStreamConstants.START_ELEMENT) {
    StartElement startElement = event.asStartElement();

    String qName = startElement.getName().getLocalPart();

    if ("book".equals(qName)) {
        book = new Book();

        // Читаем атрибут
        book.id = startElement.getAttributeByName(new QName("id")).getValue();

        catalog.books.add(book);
    }
}
// Обрабатываем текст
else if (event.getEventType() == XMLStreamConstants.CHARACTERS) {
    Characters characters = event.asCharacters();

    value.append(characters.getData());
}
```

```
// Обрабатываем закрывающий тэг
else if (event.getEventType() == XMLStreamConstants.END_ELEMENT) {
    EndElement endElement = event.asEndElement();

    String qName = endElement.getName().getLocalPart();
    String val = value.toString().trim();

    assert book != null;

    if ("author".equals(qName))
        book.author = val;
    else if ("title".equals(qName))
        book.title = val;
    else if ("genre".equals(qName))
        book.genre = val;
    else if ("price".equals(qName))
        book.price = Float.parseFloat(val);
    else if ("publish_date".equals(qName))
        book.publishDate = LocalDate.parse(val);
    else if ("description".equals(qName))
        book.description = val;

    value.setLength(0);
}
```

Аналогично происходит работа и с Cursor API:

```
// Инициализируем фабрику и Reader
XMLInputFactory factory = XMLInputFactory.newFactory();

InputStream in =
SaxExample.class.getClassLoader().getResourceAsStream("books.xml");

XMLStreamReader reader = factory.createXMLStreamReader(in);

Catalog catalog = new Catalog();
Book book = null;

StringBuilder value = new StringBuilder();

// Итерируемся по всем элементам
while (reader.hasNext()) {
    reader.next();
    // Обработчки событий
}
```

```
// Обрабатываем открывающий тэг
if (reader.getEventType() == XMLStreamReader.START_ELEMENT) {
    String name = reader.getName().getLocalPart();

    if ("book".equals(name)) {
        book = new Book();

        for (int i = 0; i < reader.getAttributeCount(); i++) {
            String attrName = reader.getAttributeName(i).getLocalPart();

            if ("id".equals(attrName)) {
                book.id = reader.getAttributeValue(i);

                break;
            }
        }

        catalog.books.add(book);
    }
}
```

```
// Обрабатываем закрывающий тэг
else if (reader.getEventType() == XMLStreamReader.END_ELEMENT) {
    String qName = reader.getName().getLocalPart();

    String val = value.toString().trim();

    assert book != null;

    if ("author".equals(qName))
        book.author = val;
    else if ("title".equals(qName))
        book.title = val;
    else if ("genre".equals(qName))
        book.genre = val;
    else if ("price".equals(qName))
        book.price = Float.parseFloat(val);
    else if ("publish_date".equals(qName))
        book.publishDate = LocalDate.parse(val);
    else if ("description".equals(qName))
        book.description = val;

    value.setLength(0);
}
// Обрабатываем текст
else if (reader.getEventType() == XMLStreamReader.CHARACTERS) {
    value.append(reader.getText());
}
```

Создается документ тоже последовательно и так же имеет два API. Пример создания с помощью Iterator API.

```
// Инициализируем фабрики и Writer
XMLOutputFactory factory = XMLOutputFactory.newFactory();
XMLEventFactory eventFactory = XMLEventFactory.newFactory();

XMLEventWriter writer = factory.createXMLEventWriter(
                                                new FileOutputStream("books.xml"));

// Указываем кодировку и версию
StartDocument startDocument = eventFactory.createStartDocument("utf-8", "1.0");
writer.add(startDocument);

// Создаем корневой тэг
StartElement catalogEl = eventFactory.createStartElement("", "", "catalog");
writer.add(catalogEl);

// Обработчики на следующем слайде

// Закрываем корневой тэг
EndElement endCatalogEl = eventFactory.createEndElement("", "", "catalog");
writer.add(endCatalogEl);

// Закрываем документ
EndDocument endDocument = eventFactory.createEndDocument();
writer.add(endDocument);

writer.flush();
writer.close();
```

```
// Добавляем подузлы
for (Book book : catalog.books) {
    // Тэг book
    StartElement bookEl = eventFactory.createStartElement("", "", "book");
    writer.add(bookEl);

    // Пишем id атрибут
    Attribute idAttr = eventFactory.createAttribute("id", book.id);
    writer.add(idAttr);

    writeContent(eventFactory, writer, "author", book.author);
    writeContent(eventFactory, writer, "title", book.title);
    writeContent(eventFactory, writer, "genre", book.genre);
    writeContent(eventFactory, writer, "price", String.valueOf(book.price));
    writeContent(eventFactory, writer, "publish_date", book.publishDate.toString());
    writeContent(eventFactory, writer, "description", book.description);

    // Закрываем тэг book
    EndElement endBookEl = eventFactory.createEndElement("", "", "book");
    writer.add(endBookEl);
}
private static void writeContent(XMLEventFactory eventFactory, XMLEventWriter
writer, String tagName, String value) throws Exception {
    StartElement startElement = eventFactory.createStartElement("", "", tagName);
    writer.add(startElement);

    Characters characters = eventFactory.createCharacters(value);
    writer.add(characters);

    EndElement endElement = eventFactory.createEndElement("", "", tagName);
    writer.add(endElement);
}
```

Cursor API даже проще:

```
XMLOutputFactory factory = XMLOutputFactory.newFactory();

XMLStreamWriter writer = factory.createXMLStreamWriter(
                                                    new FileOutputStream("books.xml"));

writer.writeStartDocument();

writer.writeStartElement("catalog");

for (Book book : catalog.books) {
    writer.writeStartElement("book");

    writeContent(writer, "author", book.author);
    writeContent(writer, "title", book.title);
    writeContent(writer, "genre", book.genre);
    writeContent(writer, "price", String.valueOf(book.price));
    writeContent(writer, "publish_date", book.publishDate.toString());
    writeContent(writer, "description", book.description);

    // book
    writer.writeEndElement();
}

// catalog
writer.writeEndElement();

writer.writeEndDocument();

writer.flush();
writer.close();
```

book.xml:

```
<?xml version="1.0" encoding="UTF-8"?><catalog>
    <book id="001">
        <author>IFMO community</author>
        <title>How to parse XML in Java</title>
        <genre>Erotic</genre>
        <price>0.0</price>
        <publish_date>2017-01-08</publish_date>
        <description>The definitive guide.</description>
    </book>
</catalog>
```