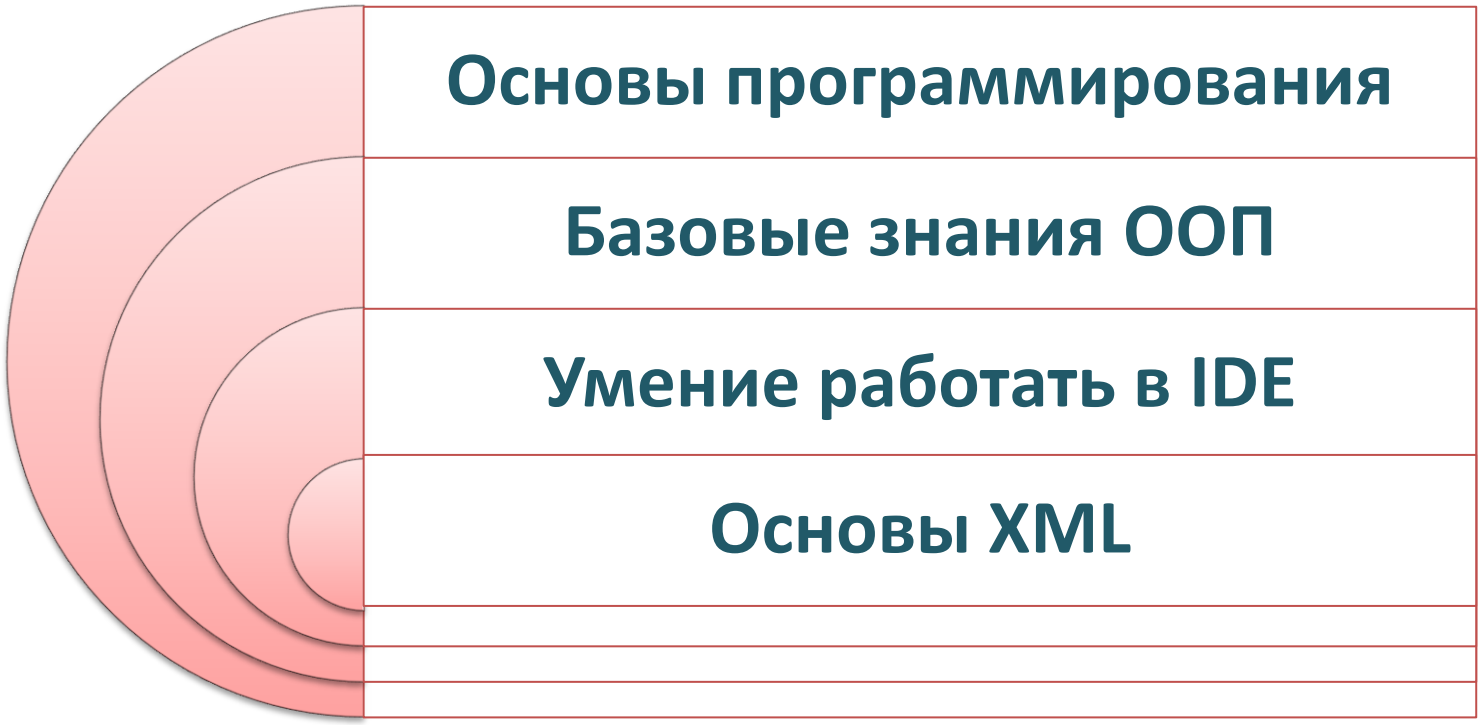
Открытый урок

Тема: Разработка бизнес приложений в Java. Урок №1. XML в Java

Герасименко Сергей Валерьевич

03 декабря 2016г.

НЕОБХОДИМЫЕ ДЛЯ УРОКА ЗНАНИЯ



Определение XML

XML — это кроссплатформенный, независящий от программного и аппаратного обеспечения инструмент передачи информации.

Отличие XML от HTML

Главное различие между XML и HTML

- ✓ XML был разработан для передачи данных.
- ✓ XML не является заменой HTML.
- ✓ XML и HTML были разработаны с различными целями:

  1) XML был создан для описания данных и основное внимание уделяется тому, какие данные передаются

  2) HTML был разработан для отображения данных основное внимание уделяются отображению данных
- ✓ Таким образом, HTML больше связан с отображением информации, в то время как XML — описанием информации.

Элемент XML документа

Теги в XML-документе не просто размечают текст — они выделяют объект, который и называется *элементом*. Элементы являются основными структурными единицами XML — именно они иерархически организуют информацию, содержащуюся в документе.

```
<myelement myattribute="myvalue">
      <mysubnode>
            sometext
       </mysubnode>
</myelement>
```

Атрибут XML-документа

В элементах можно использовать ***атрибуты*** с присвоенными им значениями.

Атрибут задается в следующем виде: **атрибут = "значение"**

Например, в записи гипертекстовой ссылки

Поиск информации

элемент **a** имеет атрибут **href**, которому присвоено значение http://www.ya.ru".

В языке XML атрибуты всегда должны иметь значения!

Обобщение

XML документ состоит из тэгов. Каждому открывающему тэгу соответствует закрывающий тэг. Внутри тэга могут быть вложенные тэги. У тэгов могут быть атрибуты. У тэга обязательно должно быть какое-то значение. Например просто текст.

```
<tag>
   <child attr="value">
      text
   </child>
</tag>
```

Пример XML документа

Заголовок XML документа:

<?xml version="1.1" encoding="UTF-8" **?>**
или
<?xml version="1.0" encoding="windows-1251"**?>**

```
<?xml version="1.0" encoding="WINDOWS-1251"?>
<tutorial>
    <title>"Заметки об XML"</title>
    <author>Леонов Игорь Васильевич</author>
</tutorial>
```

Описание данных XML

Для описания данных XML можно использовать DTD (Document Type Definition) или схемы XML. Данные механизмы описывают документ и говорят какие в нем должны быть атрибуты, тэги, в какой последовательности они должны идти и сколько раз они должны встречаться. DTD описание встречается редко т.к. этот механизм очень ограничен по функциональности - он может задать последовательности тэгов, сколько раз встречаются тэги.

Схема уже накладывает более жесткие ограничения вплоть до того - какого типа могут быть атрибуты. Какого типа могут быть значения. Существует средства которые позволяют автоматически проверять XML документы на соответствие схеме. Schema это xml документ который описывает другой xml документ. Существуют редакторы которые автоматически создают схемы.

Типы XML-файлов

1) Правильные
2) Неправильные
3) Правильно-форматированные

Практика

Создать xml документ, содержащий информацию о трех книгах по программированию

Обработка XML документа

Наиболее общей задачей обработки XML является **разбор (parsing)** XML-документа.

XML-парсер - это часть кода, которая читает XML-документ и анализирует его структуру.

Как использовать парсер

- Создать объект парсера
- Указать объекту парсера на ваш XML-документ
- Обработать результаты

Виды XML-парсеров

1) Document Object Model (DOM), Уровня 2

2) Simple API for XML (SAX), Версия 2.0

3) XPath (XML Path Language) — язык запросов к элементам XML-документа

Интерфейс DOM

DOM - это «интерфейс» для работы со структурой XML-документов. Когда вы разбираете XML-документ при помощи парсера DOM, вы получаете иерархическую структуру данных (дерево DOM), которая представляет все, что парсер нашел в XML-документе. Вы можете затем использовать функции DOM, чтобы манипулировать деревом, перемещать ветви, добавлять новые ветви, удалять части дерева. Парсер DOM возвращает вам структуру дерева; это дерево DOM содержит много Node (узлов). С точки зрения языка Java, Node - это интерфейс. Node является основным типом данных DOM; все в дереве DOM является Node того или иного типа. DOM берет весь документ и создает иерархическую структуру и связывает ее между собой.

То есть мы получаем сразу иерархию всего документа.

Алгоритм работы DOM

1) Из DOM создать фабрику и из фабрики создать парсер. Фабрика поднимает все необходимые классы и ресурсы.
2) Создать объект парсера из фабрики. Document doc = d.parse(file); этот объект хранит уже полностью всю структуру. Документ это класс, описывающий структуру xml и внутри него лежит много элементов. У каждого элемента есть методы. Например, чтобы получить потомка используется getChild(),

получить атрибут – getAttribute()

3) Element e = doc.getDocumentElement() - для получения root элемента
4) Теперь можем обращаться к этому элементу

e.getElementsByTagName(String tag);

5) Если у элемента есть текст то чтобы его получить используем метод e.getElementsByTagName(String child).item(0).getNodeValue();

Функции DOM

- **Document.getDocumentElement():** Возвращает корень дерева DOM. (Эта функция является методом интерфейса Document; он не определен для других подтипов Node.)
- **Node.getFirstChild() и Node.getLastChild():** Возвращают первого или последнего потомzка данного Node.
- **Node.getNextSibling() и Node.getPreviousSibling**(): Возвращают следующего или предыдущего "брата" данного Node.
- **Element.getAttribute(String attrName):** Для данного Element возвращает значение атрибута с именем attrName. Если вам нужно значение атрибута "id", используйте Element.getAttribute("id"). Если этот атрибут не существует, метод возвращает пустую строку ("").

Пример

- Создать объект парсера
- Указать объекту парсера разобрать XML-файл
- Обойти результирующее дерево DOM и напечатать содержимое различных узлов в стандартный вывод

SAX (англ. *«Simple API for XML»*)

- SAX – способ последовательного чтения/записи XML-файлов: вы создаете парсер SAX, и парсер сообщает вам (он проталкивает вам события), когда он находит в XML-документе разные вещи. В частности, вот как SAX решает вопросы, упомянутые выше:
- SAX не строит в памяти дерево XML-документа. Парсер SAX посылает вам события по мере того, как он находит в XML-документе разные вещи. На ваше усмотрение решать, как (и если) вы хотите сохранять эти данные.
- Парсер SAX не создает никаких объектов. Вы можете на выбор создавать объекты, если вы хотите, но это - ваше решение, а не парсера.
- SAX начинает посылать вам события немедленно. Вы не должны ожидать, пока парсер закончит чтение документа.

Алгоритм работы SAX

1) Создаем SAXParserFactory f = SAXParserFactory.newInstance(); //Обращаемся к фабрике, которая содержит всю информацию по парсеру.

2) У фабрики запрашиваем SAX парсер: SAXParser p = f.newSaxParser(); //теперь парсер есть

3) Теперь нашему парсеру нужно передать наш XML документ. Для этого используется

p.parse(xml документ, handler)Но прежде чем вызвать этот метод, ему нужно установить handler, т.е. ваш обработчик, который будет слушать и принимать сообщения от парсера

Разберем слушателя этих сообщений, т.е. handler.

4) Существует класс DefaultHandler. Мы должны написать класс, который расширяет данный.

DefaultHandler h = new DefaultHandler(){Здесь напишем реализацию нашего интерфейса} У этого интерфейса есть несколько методов

а)void startElement (String uri, String name, String qname, Attributes atr) Данный метод будет вызван, когда SAX парсер найдет первый элемент (тэг).

здесь name - это имя элемента, qname - имя атрибута, Attributes это класс который содержит все атрибуты данного тэга.

Методы нужно менять самому под свои потребности. То есть парсер просто находит элементы.

б)void endDocument(String uri, String name, String qname, Attributes atr)

в)void characters(char[]ch,int start, int length) - этот метод говорит, что нашел набор символов. ch - это массив который содержит все символы которые он нашел. Чтобы из массива ch получить нужные значения нужно использовать start и length.

Модификатор и Тип	Метод и Описание
void	characters(char[] ch, int start, int length) Получите уведомление о символьных данных в элементе.
void	endDocument() Получите уведомление о конце документа.
void	endElement(String uri, String localName, String qName) Получите уведомление о конце элемента.
void	endPrefixMapping(String prefix) Получите уведомление о конце отображения Пространства имен.
void	error(SAXParseException e) Получите class in org.xml.sax станавливаемой ошибке синтаксического анализатора.
void	fatalError(SAXParseException e) Сообщите о фатальном XML, анализирующем ошибку.
void	ignorableWhitespace(char[] ch, int start, int length) Получите уведомление об игнорируемом пробеле в контенте элемента.
void	notationDecl(String name, String publicId, String systemId) Получите уведомление об объявлении нотации.
void	processingInstruction(String target, String data) Получите уведомление об инструкции обработки.
InputSource	resolveEntity(String publicId, String systemId) Разрешите внешний объект.
void	setDocumentLocator(Locator locator) Получите объект Локатора для событий документа.
void	skippedEntity(String name) Получите уведомление о пропущенном объекте.
void	startDocument() Получите уведомление о начале документа.
void	startElement(String uri, String localName, String qName, Attributes attributes) Получите уведомление о запуске элемента.
void	startPrefixMapping(String prefix, String uri) Получите уведомление о запуске отображения Пространства имен.
void	unparsedEntityDecl(String name, String publicId, String systemId, String notationName) Получите уведомление о непроанализированном объявлении объекта.
void	warning(SAXParseException e) Получите уведомление о предупреждении синтаксического анализатора.

XPath (XML Path Language)

XPath (XML Path Language) — язык запросов к элементам XML-документа. XPath призван реализовать навигацию по DOM в XML. В XPath используется компактный синтаксис, отличный от принятого в XML.

```
public static void main(String[] args)
 throws ParserConfigurationException, SAXException,
        IOException, XPathExpressionException {

  XPathFactory factory = XPathFactory.newInstance();
  XPath xpath = factory.newXPath();
  XPathExpression expr
   = xpath.compile("//book[author='Neal Stephenson']/title/text()");

  Object result = expr.evaluate(new InputSource(new FileReader("book.xml")),
                                XPathConstants.NODESET);
  NodeList nodes = (NodeList) result;
  for (int i = 0; i < nodes.getLength(); i++) {
      System.out.println(nodes.item(i).getNodeValue());
  }

}
```

XPath (XML Path Language)

/bookstore/book[1]
/bookstore/book[last()]
/bookstore/book[last()-1]
/bookstore/book[position()<3]
//title[@lang]
//title[@lang='en']
/bookstore/book[price>35.00]
/bookstore/book[price>35.00]/title

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

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

Благодарю за внимание!