# day06-XML解析器

## 解析器概述

###### 什么是解析器

　　XML是保存数据的文件，XML中保存的数据也需要被程序读取然后使用。那么程序使用什么来读取XML文件中的数据呢？XML解析器！例如.properties文件的解析器是Properties类一样！

　　XML不只被Java语言使用，还被C++、C#、Javascript等等语言使用，所以解析XML不是一门语言的工作！

###### 2　DOM和SAX介绍

主流的XML解析有两种标准：DOM和SAX。它们是标准，是思想，不是真正的解析器，它们是跨语言的！！！

* DOM（Document Object Model）：W3C组织提供的解析XML文档的标准接口；
* SAX（Simple API for XML）：(草民)社区讨论的产物，是一种事实上的标准。

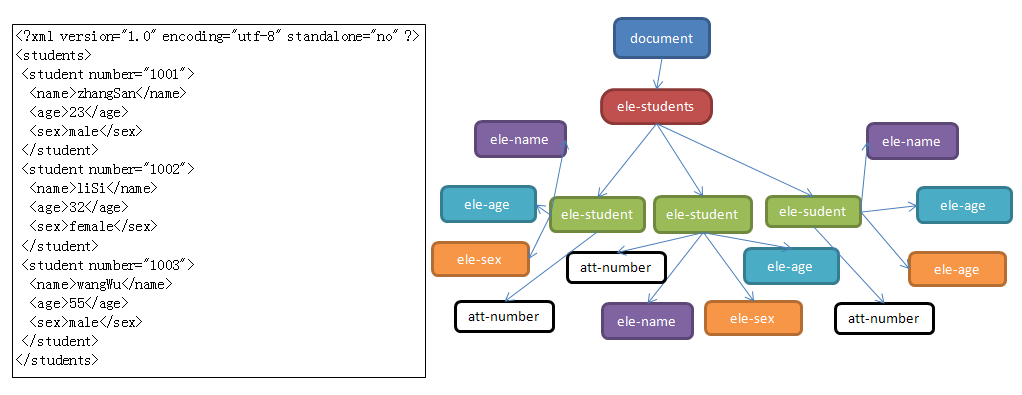
Apache的xerces组件实现了DOM和SAX，所以在我们在Java中解析XML需要使用xerces(JDK提供了)。所以我们称xerces是DOM、SAX解析器。

2.1　DOM解析原理

DOM要求解析器把整个XML文档装载到一个Document对象中。即使用DOM解析器解析XML文档的结果就是一个Document对象。

一个XML文档解析后对应一个Document对象，可以通过Document对象获取根元素，然后在通过根元素获取根元素的子元素…，这说明DOM解析方式保留了元素之间的结构关系(DOM树)！

* 优点：元素与元素之间的结构关系保留了下来；
* 缺点：如果XML文档过大，那么把整个XML文档装载进内存，可能会出现内存溢出的现象！



2.2　SAX解析原理

DOM解析后的结果是一个Document对象，而SAX解析没有结果！SAX要求在开始解析之前用户提供一个接口的实现对象，然后把接口实现对象传递给SAX解析器，然后在SAX解析器的过程中不调用实现对象的方法。

* DOM是解析时把数据放到了Document对象中，然后用户从Document中获取需要的数据；
* SAX要求用户参与到解析过程中来，把想要做的事情写到接口实现对象中，然后SAX在解析过程中来调用接口实现对象的方法。

你看过三国么？听说过三气周瑜么？其中二气周瑜你知道么？故事情节大致如下：

倒霉的周瑜让孙权把自己的妹妹嫁给刘备，让刘备来吴国完婚！目的是把刘备软禁在吴国，或者把刘备干掉。如果刘备不同意，那么孙权就有了打刘备，抢荆州的借口了。

孔明识破了这一计，最终让刘备与赵云去吴国。但孔明给了赵云三个锦囊：

* 到了吴国南徐开第一个锦囊；
* 在吴国住到年终开第二个锦囊；
* 回家途中被吴军阻拦开第三个锦囊。

锦囊的内容是什么呢？赵云和刘备性命如何呢？诸葛亮又是如何二气周瑜的呢？这里没有了，自己去买本《三国演义》看吧！

其中赵云就是SAX解析器，赵负责去吴国，SAX负责解析XML；

其中孔明的三个锦囊就是接口实现中的三个方法；

赵云会在特定时刻打开锦囊，依计而行，SAX解析器会在解析过程中特定时刻调用接口实现中的某一方法。

取亲之旅 🡨🡪 被解析的XML

赵云 🡨🡪 SAX解析器

锦囊 🡨🡪接口在三个方法（由我们完成）

赵云在取亲之旅中，会在发生特定事件时，执行特定的锦囊。例如在到达吴国南徐时，执行第一个锦囊上的妙计。

SAX解析器会在解析XML文档的过程中，在发生特定事件时，调用接口中特定的方法。例如在SAX解析到某个元素的开始标签时，输出元素名称！其中解析到开始标签就是特定的事件，而输出元素名称，就是接口中方法的实现。

接口中方法如下：

|  |
| --- |
| **public** **void** startDocument()**;**  **public** **void** endDocument();  **public** **void** startElement(String uri, String localName, String qName,  Attributes atts);  **public** **void** endElement(String uri, String localName, String qName)  **public** **void** characters(**char**[] ch, **int** start, **int** length);  **public** **void** ignorableWhitespace(**char**[] ch, **int** start, **int** length);  **public** **void** processingInstruction(String target, String data); |

　　接口的实现由我们来完成，然后我们需要把接口实现类对象“交给”SAX解析器，然后让SAX开始解析。SAX会在特定事件发生时，调用接口中的方法，完成我们交给它的任务。

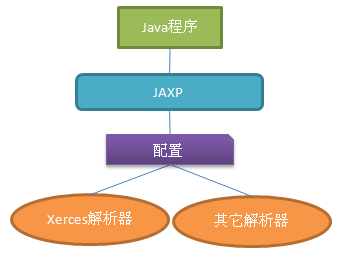
* 优点：适合解析大XML文件（内存空间占用小），因为是解析一行处理一行，处理完了就不需要在保留数据了；
* 缺点：因为是解析一行处理一行，解析之后数据就丢失了，所以元素与元素之间的结构关系没有保留下来。

###### 3　JAXP介绍(不重要)

我们知道有很多像xerces一样的解析器，都对DOM和SAX提供了实现，那么如果我们在项目中一开始使用了解析A，然后因为某些原因想更换成解析B，那么就需要修改项目。

JAXP（Java API for XML Processing）是由Java提供的，JAXP是对所有像xerces一样的解析的提供统一接口的API。

当我们使用JAXP完成解析工作时，还需要为JAXP指定xerces或其他解析器，当需要更换解析器时，无需修改代码，只需要修改配置即可。



　　JAXP不是解析器，但使用它可以方便的切换解析器。所以在我们的程序中只会使用JAXP，而不会直接使用Xeces。

###### 4　JDOM和DOM4j介绍

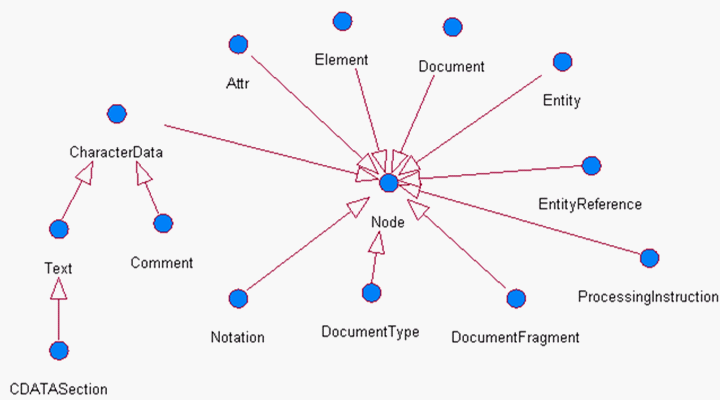
DOM和SAX是跨语言的XML解析准备，在Java中使用并不方便。而JDOM和DOM4j是专门为Java语言提供的解析工具！使用起来很方便，所以真实开发中使用JDOM或DOM4J比较多。

又因为DOM4J与JDOM比较结果为DOM4j完胜，所以我们这里只会对DOM4j介绍，而不会介绍JDOM。

## DOM和SAX解析

###### 1　DOM结构模型

DOM中的核心概念就是节点，在XML文档中的元素、属性、文本、处理指令，在DOM中都是节点！



###### 2　JAXP之DOM解析

使用DOM解析XML的目标就是获取到Document对象，然后在从Document中获取到需要的数据。Document对象就是XML文档在内存中的样子。

1. 获取Document三步：

|  |
| --- |
| DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();  DocumentBuilder builder = factory.newDocumentBuilder();  Document document = builder.parse(new File("students.xml")); |

1. 遍历Document：

|  |
| --- |
| @Test  **public** **void** fun1() **throws** SAXException, IOException, ParserConfigurationException {  DocumentBuilderFactory factory = DocumentBuilderFactory.*newInstance*();  DocumentBuilder builder = factory.newDocumentBuilder();  Document document = builder.parse(**new** File("src/students.xml"));    // 获取根元素  Element rootEle = document.getDocumentElement();  // 获取元素的所有子节点  NodeList nodeList = rootEle.getChildNodes();  // 循环遍历所有节点  **for**(**int** i = 0; i < nodeList.getLength(); i++) {  // 获取其中每个节点  Node node = nodeList.item(i);  // 判断节点的类型是否为元素  **if**(node.getNodeType() == Node.*ELEMENT\_NODE*) {  // 强转成元素类型  Element stuEle = (Element) node;  // 获取元素的名称  String eleName = stuEle.getNodeName();  // 获取元素的number属性值  String number = stuEle.getAttribute("number");  // 获取名为name的子元素，因为返回值为NodeList，  // 所以需要使用item(0)方法获取第一个name子元素  // getTextContent()是获取节点的文本内容  String name = stuEle.getElementsByTagName("name").item(0).getTextContent();  String age = stuEle.getElementsByTagName("age").item(0).getTextContent();  String sex = stuEle.getElementsByTagName("sex").item(0).getTextContent();    System.*out*.println(eleName + ":[number=" + number + ", name=" + name + ", age=" + age + ", sex=" + sex + "]");  }  }  } |

###### 3　JAXP之SAX解析

使用SAX解析XML文档需要先给出DefualtHandler的子类，重写其中的方法。然后在使用SAX开始解析时把DefaultHandler子类对象传递给SAX解析器。

|  |
| --- |
| **class** MyContentHandler **extends** DefaultHandler {  **public** **void** startDocument() **throws** SAXException {  System.*out*.println("开始解析...");  }  **public** **void** endDocument() **throws** SAXException {  System.*out*.println("解析结束...");  }  **public** **void** startElement(String uri, String localName, String qName,  Attributes atts) **throws** SAXException {  System.*out*.println(qName + "元素解析开始");  }  **public** **void** endElement(String uri, String localName, String qName)  **throws** SAXException {  System.*out*.println(qName + "元素解析结束");  }  **public** **void** characters(**char**[] ch, **int** start, **int** length)  **throws** SAXException {  System.*out*.print(**new** String(ch, start, length).trim());  }  } |

使用SAX解析首先需要获取工厂，再通过工厂获取解析器对象，然后使用解析对象完成解析工作：

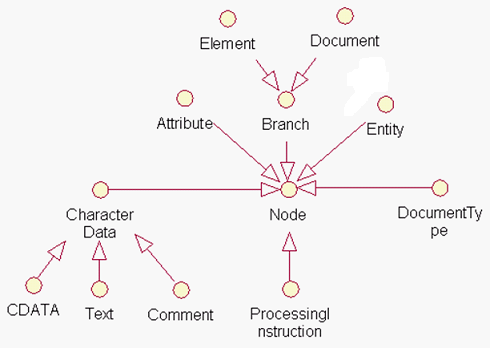
|  |
| --- |
| SAXParserFactory factory = SAXParserFactory.*newInstance*();  SAXParser parser = factory.newSAXParser();  parser.parse(**new** File("src/students.xml"), **new** MyContentHandler()); |

## DOM4J解析

###### 1　DOM4J概述

DOM4J是针对Java开发人员专门提供的XML文档解析规范，它不同与DOM，但与DOM相似。DOM4J针对Java开发人员而设计，所以对于Java开发人员来说，使用DOM4J要比使用DOM更加方便。

在DOM4J中，也有Node、Document、Element等接口，结构上与DOM中的接口比较相似。但它们是不同的类：



* Node
* Attribute：表示属性节点；
* Branch：表示可以包含子元素的节点：
* Document：表示整个文档；
* Element：表示元素节点；
* CharacterData：表示文本节点：
* Text：表示文本内容；
* CDATA：表示CDATA段内容；
* Comment：表示注释内容。

　　我们再次强调，DOM和DOM4J是不同的，DOM中的Document是org.w3c.Document，而DOM4J中的Document是org.dom4j.Document，它们是不同的类，其他Node、Element也是一样。

###### 2　读取、保存、创建Document

使用dom4j需要导入：

* dom4j.jar
* jaxen.jar

1. 读取

|  |
| --- |
| SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml"); |

1. 保存XML文档

|  |
| --- |
| // 创建格式化器，使用\t缩进，添加换行  OutputFormat format = **new** OutputFormat("\t", **true**);  // 清空数据中原有的换行  format.setTrimText(**true**);  // 创建XML输出流对象  XMLWriter writer = **new** XMLWriter(**new** FileWriter("src/a.xml"), format);  // 输出Document  writer.write(doc);  // 关闭流  writer.close(); |

1. 创建Document

|  |
| --- |
| Document doc = DocumentHelper.*createDocument*(); |

###### 3　遍历Document

|  |
| --- |
| **public** **void** fun1() **throws** DocumentException {  SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml");  // 获取根元素  Element rootEle = doc.getRootElement();  // 获取根元素的所有子元素  List<Element> eleList = rootEle.elements();  // 遍历元素集合  **for** (Element stuEle : eleList) {  // 获取元素名称  String eleName = stuEle.getName();  // 获取元素的number属性值  String number = stuEle.attributeValue("number");  // 获取元素的name子元素内容  String name = stuEle.elementText("name");  // 获取元素的age子元素内容  String age = stuEle.elementText("age");  // 获取元素的sex子元素内容  String sex = stuEle.elementText("sex");  System.*out*.println(eleName + ": [number=" + number + ", name="  + name + ", age=" + age + ", sex=" + sex + "]");  }  } |

###### 4　添加student元素

|  |
| --- |
| SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml");    // 获取根元素<students>  Element root = doc.getRootElement();  // 为root添加名为student的子元素，并返回这个新添加的子元素  Element stuEle = root.addElement("student");  // 给元素添加属性number，值为123  stuEle.addAttribute("number", "123");  // 添加子元素name，并设置name子元素的文本内容为wangWu  stuEle.addElement("name").setText("wangWu");  stuEle.addElement("age").setText("30");  stuEle.addElement("sex").setText("male");    //////////////////    // 创建格式化器，使用\t缩进，添加换行  OutputFormat format = **new** OutputFormat("\t", **true**);  // 清空数据中原有的换行  format.setTrimText(**true**);  // 创建XML输出流对象  XMLWriter writer = **new** XMLWriter(**new** FileWriter("src/a.xml"), format);  // 输出Document  writer.write(doc);  // 关闭流  writer.close(); |

###### 5　查询元素

|  |
| --- |
| SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml");    /\*  \* selectSingNode()方法的参数是XPath  \* XPath是在XML文档中查找的一门表达式语言  \* “//”表示查找整个XML文档  \* student表示查找名为student的元素  \* []表示条件  \* @number表示number属性  \* @number='ITCAST\_1001'表示条件为number属性等于ITCAST\_1001  \*  \* selectSingNode()方法在查找到多个满足XPath的元素时，只返回第一个。  \*/  Element stuEle1 = (Element)doc.selectSingleNode("//student[@number='ITCAST\_1001']");  // 把元素转换成字符串  System.*out*.println(stuEle1.asXML());    // 查找name子元素内容为liSi的student元素  Element stuEle2 = (Element) doc.selectSingleNode("//student[name='liSi']");  System.*out*.println(stuEle2.asXML()); |

###### 6　修改元素

|  |
| --- |
| SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml");    // 查找元素  Element stuEle = (Element)doc.selectSingleNode("//student[@number='ITCAST\_1001']");  // 修改student元素的name子元素内容为“张三”  stuEle.element("name").setText("张三");    //////////////////    // 创建格式化器，使用\t缩进，添加换行  OutputFormat format = **new** OutputFormat("\t", **true**);  // 清空数据中原有的换行  format.setTrimText(**true**);  // 创建XML输出流对象  XMLWriter writer = **new** XMLWriter(**new** FileWriter("src/a.xml"), format);  // 输出Document  writer.write(doc);  // 关闭流  writer.close(); |

###### 7　删除学生元素

|  |
| --- |
| SAXReader reader = **new** SAXReader();  Document doc = reader.read("src/students.xml");    // 查找元素  Element stuEle = (Element)doc.selectSingleNode("//student[@number='ITCAST\_1001']");  // 获取父元素来删除元素  stuEle.getParent().remove(stuEle);    //////////////////    // 创建格式化器，使用\t缩进，添加换行  OutputFormat format = **new** OutputFormat("\t", **true**);  // 清空数据中原有的换行  format.setTrimText(**true**);  // 创建XML输出流对象  XMLWriter writer = **new** XMLWriter(**new** FileWriter("src/a.xml"), format);  // 输出Document  writer.write(doc);  // 关闭流  writer.close(); |

###### 8　dom4j API介绍

Node方法：

* String asXML()：把当前节点转换成字符串，如果当前Node是Document，那么就会把整个XML文档返回；
* String getName()：获取当前节点名字；Document的名字就是绑定的XML文档的路径；Element的名字就是元素名称；Attribute的名字就是属性名；
* Document getDocument()：返回当前节点所在的Document对象；
* short getNodeType()：获取当前节点的类型；
* String getNodeTypeName()：获取当前节点的类型名称，例如当前节点是Document的话，那么该方法返回Document；
* String getStringValue()：获取当前节点的子孙节点中所有文本内容连接成的字符串；
* String getText()：获取当前节点的文本内容。如果当前节点是Text等文本节点，那么本方法返回文本内容；例如当前节点是Element，那么当前节点的内容不是子元素，而是纯文本内容，那么返回文本内容，否则返回空字符串；
* void setDocument(Document doc)：给当前节点设置文档元素；
* void setParent(Element parent)：给当前节点设置父元素；
* void setText(String text)：给当前节点设置文本内容；

Branch方法：

* void add(Element e)：添加子元素；
* void add(Node node)：添加子节点；
* void add(Comment comment)：添加注释；
* Element addElement(String eleName)：通过名字添加子元素，返回值为子元素对象；
* void clearContent()：清空所有子内容；
* List content()：获取所有子内容，与获取所有子元素的区别是，<name>liSi</name>元素没有子元素，但有子内容；
* Element elementById(String id)：如果元素有名为“ID”的属性，那么可以使用这个方法来查找；
* int indexOf(Node node)：查找子节点在子节点列表中的下标位置；
* Node node(int index)：通过下标获取子节点；
* int nodeCount()：获取子节点的个数；
* Iterator nodeIterator()：获取子节点列表的迭代器对象；
* boolean remove(Node node)：移除指定子节点；
* boolean remove(Commont commont)：移除指定注释；
* boolean remove(Element e)：移除指定子元素；
* void setContent(List content) ：设置子节点内容；

Document方法：

* Element getRootElement()：获取根元素；
* void setRootElement()：设置根元素；
* String getXmlEncoding()：获取XML文档的编码；
* void setXmlEncoding()：设置XML文档的编码；

Element方法：

* void add(Attribute attr)：添加属性节点；
* void add(CDATA cdata)：添加CDATA段节点；
* void add(Text Text)：添加Text节点；
* Element addAttribute(String name, String value)：添加属性，返回值为当前元素本身；
* Element addCDATA(String cdata)：添加CDATA段节点；
* Element addComment(String comment)：添加属性节点；
* Element addText(String text)：添加Text节点；
* void appendAttributes(Element e)：把参数元素e的所有属性添加到当前元素中；
* Attribute attribute(int index)：获取指定下标位置上的属性对象；
* Attribute attribute(String name)：通过指定属性名称获取属性对象；
* int attributeCount()：获取属性个数；
* Iterator attributeIterator()：获取当前元素属性集合的迭代器；
* List attributes()：获取当前元素的属性集合；
* String attributeValue(String name)：获取当前元素指定名称的属性值；
* Element createCopy()：clone当前元素对象，但不会copy父元素。也就是说新元素没有父元素，但有子元素；
* Element element(String name)：获取当前元素第一个名称为name的子元素；
* Iterator elementIterator()：获取当前元素的子元素集合的迭代器；
* Iterator elementIterator(String name)：获取当前元素中指定名称的子元素集合的迭代器；
* List elements()：获取当前元素子元素集合；
* List elements(String name)：获取当前元素指定名称的子元素集合；
* String elementText(String name)：获取当前元素指定名称的第一个元素文件内容；
* String elementTextTrime(String name)：同上，只是去除了无用空白；
* boolean isTextOnly()：当前元素是否为纯文本内容元素；
* boolean remove(Attribute attr)：移除属性；
* boolean remove(CDATA cdata)：移除CDATA；
* boolean remove(Text text)：移除Text。

DocumentHelper静态方法介绍：

* static Document createDocument()：创建Dcoument对象；
* static Element createElement(String name)：创建指定名称的元素对象；
* static Attribute createAttrbute(Element owner, String name, String value)：创建属性对象；
* static Text createText(String text)：创建属性对象；
* static Document parseText(String text)：通过给定的字符串生成Document对象；