# XML

### 1.xml的基本知识

xml：extensible markup language(扩展性标记语言)。其作用：**存储和传输数据的，一般用来当配置文件**。xml文件中区分大小写。xml文件是以“.xml”为扩展名的文件。

xml文件开头要加上版本信息和编码格式以及声明为xml文件。<?xml version="1.0" encoding="utf-8"?>，尖括号中的小写xml声明了这是一个xml文件，不能写成大写的。

一份xml文件中必须有唯一且单一的根元素。每一个元素都必须有一个开始和结束标签(除了空标签）。标签名和属性名必须合法，大小写看作不同。属性的值必须以引号括起来，单双引号都可以，多个属性之间用空格隔开。

在java程序中读取xml文件的过程也称为解析xml文件。有四种解析方式：**DOM、SAX、DOM4J、JDOM**。其中DOM和SAX是属于java解析xml的官方方式，不需要下载jar包。而DOM4J和JDOM则属于第三方，需要网上下载jar包然后导入到项目中。

**常用的节点类型：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **节点类型** | **Named Constant** | **getNodeName**  **的返回值** | **getNodeValue**  **的返回值** | **getNodeType**  **的返回值** |
| Element | ***ELEMENT\_NODE*** | element name | **null** | **1** |
| Attr | ***ATTRIBUTE\_NODE*** | 属性名称 | 属性值 | **2** |
| Text | ***TEXT\_NODE*** | #text | 节点内容 | **3** |

xml文件中，**空格与换行符被看成是一个文本节点，起始标签和结束标签看成一个元素节点**。

XML文档是一个自我描述的数据集合。优点：自我描述性、可移植性、以树结构描述数据、提供可用于数据库的功能。缺点：数据存取慢、缺少数据库的功能、在生产环境中将失效 。

## 一、DOM

DOM(Document Object Model)：文档对象模型

Document导包：**import** org.w3c.dom.Document;

Element导包：**import** org.w3c.dom.Element;

其他的导包：**import** javax.xml.\*;

注意在**使用DOM生成xml文件时，创建Document对象使用的是newDocument方法；解析xml文件时，创建Document对象使用的是parse(File file)方法**。

DOM最适用的情形：在结构上修改 XML 文档时，在内存中与其他应用程序共享文档时。

DOM的缺点：将整个文档存储在内存中，方法命名惯例不符合 Java 编程惯例。

### 1.DOM生成xml文件并添加数据

#### ⑴常用的方法

**DocumentBuilderFactory**：文档构建器工厂

**DocumentBuilder**：文档构建器

**TransformerFactory**：传输工厂

Document**.createElement(String tagName)**：创建节点，参数为节点的名称。返回值为Element类型。

Node**.setTextContent(String textContent)**：为节点设置内容，无返回值。

Element**.setAttribute(String name,String value)**：为节点设置属性名和属性值。

Transformer**.transform(Source xmlSource，Result outputTarget)**：将Source类型的xmlSource文件即源文件传输到目标中，即Result类型的outputTarget中。

#### ⑵步骤

**1.准备工作：创建三个对象**

(1)创建DocumentBuilderFactory对象：自身的newInstance()方法创建

dBFactory=DocumentBuilderFactory.*newInstance*();

(2)创建DocumentBuilder对象：解析器工厂的newDocumentBuilder()方法创建

dBuilder = dBFactory.newDocumentBuilder();

(3)创建Document对象：调用的是DocumentBuilder的newDocument()方法

Document document=dBuilder.newDocument();

上面也可以直接写成一个的形式，如下：

Document document = DocumentBuilderFactory.*newInstance*(). newDocument Builder() . newDocument();

**2. 生成xml文件:**

(1)创建根节点：

Element students=document.createElement("Students");

(2)将根节点添加到document中：

document.appendChild(students);

(3)采用for循环创建节点：

创建父结点，同时为父节点设置属性名和属性值

Element student=document.createElement("Student");

student.setAttribute("id", ""+i);

创建子节点，同时为子节点赋值

Element name=document.createElement("name");

name.setTextContent("张"+i);

将子节点添加到父节点中

student.appendChild(name);

将父结点添加到根节点中

students.appendChild(student);

**3.利用传输器将document内存树传送到文件中**

(1)创建传输工厂的对象：自身的newInstance()方法创建

tsfFactory=TransformerFactory.*newInstance*();

(2)创建传输器对象：调用传输工厂的newTransformer()方法创建

tsf=tsfFactory.newTransformer();

(3)创建传输过程的两个对象

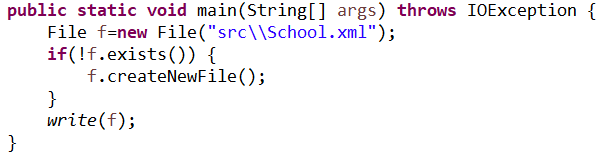
Source xmlSource=**new** DOMSource(document);

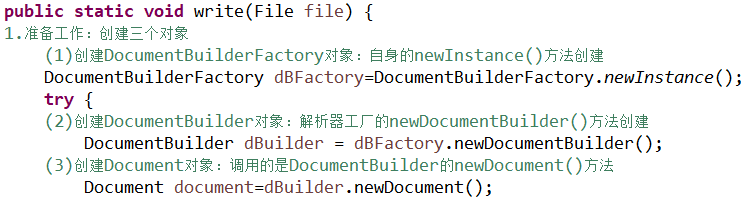
Result outTarget=**new** StreamResult(file);

(4)将document内存树传输到指定文件中

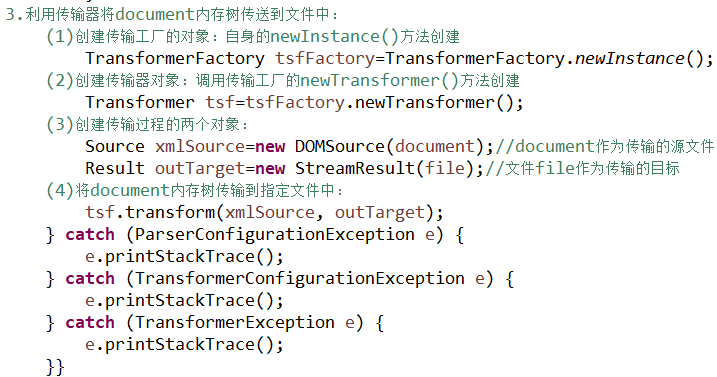
tsf.transform(xmlSource, outTarget);

#### ⑶实例：









### 2.DOM解析xml文件获取数据。

#### ⑴常用的方法

DocumentBuilder**.parse(File f)**：返回值为Document，参数为要解析的xml文件。

Document**.getElementsByTagName(String tagname)**：返回值为NodeList，参数为标签名。

NodeList**.getLength()**：返回值为int类型，获得节点集合的长度。

NodeList**.item(int index)**：返回值为Node类型，即返回集合中索引位置上的节点。

Node**.getAttributes()**：返回值类型为NamedNodeMap，返回的是节点的所有属性的集合。

Node**.getNodeName()**：返回值String类型。获得节点的名称。

Node**.getNodeValue()**：返回值String类型。获得节点的值。

Node**.getNodeType()**：判断节点的类型，Node.***ELEMENT\_NODE***指的是元素节点，NODE.***ATTRIBUTE\_NODE***是属性节点，NODE.T***EXT\_NODE***是文本节点。返回值为short类型。

Node**.getChildNodes(**)：返回值为NodeList，获得子节点的集合。

Element**.getAttribute(String name)**：返回值为String类型，获得节点的属性。参数为要检索的属性名。

Element**.getTagName()**：返回值为String类型，获得元素节点的标签名。

Node**.getTextContent()**：返回值为String类型。获得节点的文本内容。这个是Node的方法。因为Element接口继承了Node接口，所以Element可以使用。

**Element接口继承了Node接口。**

Node**.getTextContent()**：返回值为String类型，获得节点存放的内容。

Node**.getFirstChild()**：返回值为Node，返回的是第一个子节点，即根节点，若没有则返回null。

Node**.getLastChild()**：返回值为Node，返回的是最后一个子节点。

#### ⑵步骤

**1.准备工作：获得三个对象**

**⑴创建文档构建器工厂的对象：**

dBFactory**=DocumentBuilderFactory.newInstance()**

**⑵创建文档构建器对象：**

dBuilder=dBFactory**.newDocumentBuilder()**

**⑶用文档构建器的parse()解析xml文件，并获得Document对象：**

**Document** document=dBuilder.**parse(File file);**

**2.解析xml文件**

**2.1获得节点的集合并遍历：有两种情况**

**方法一：通过标签名来获得节点的集合是随机访问的方式。**

NodeList bookList=document.getElementsByTagName("Book");

**方法二：通过根节点来获得节点集合。通过getFirstChild()获得根节点并强转为Element，然后在通过getChildNodes()获得根节点下的子节点集合。这种方式需要注意：**[**例3有详细说明**](#_③从根节点开始遍历且单个属性实例3(重点)：)

Element bookShelf = (Element) document.getFirstChild();

NodeList bookList = bookShelf.getChildNodes();

**2.2使用DOM解析xml文件的属性名和属性值：有两种情况**

**⑴当未知节点属性的个数和属性名时：**

**获得节点的所有属性集合，遍历集合，并通过getNodeName()和getNodeValue()获得属性名和属性值。**

NamedNodeMap bookAttrList = book.getAttributes();

**⑵已知节点只有一个属性，如id属性：**

**在遍历NodeList类型的集合时，调用item()获得节点，将这个节点强制转换成Element类型。然后调用Element的getAttribute(String name)方法或者属性值。**



**2.3使用DOM解析xml文件的节点名和节点值：**

**⑴获得子节点的集合：getChildNodes()**

NodeList child = book.getChildNodes();

**⑵遍历子节点的集合：**

**①采用NodeList的item()方法得到子节点：**

Node n=child.item(x);

**②判断节点的类型：getNodeType()，Node.ELEMENT\_NODE为元素标签。**

**if**(n.getNodeType()==Node.***ELEMENT\_NODE***)

**③获得节点名和节点值：多种方法。**

**方法1：强转为Element类型，然后调用Element类的getTagName()、getTextContent()方法。**

**方法2：不对Node进行强转。直接调用Node类的getNodeName()获得节点名。然后再调用Node类的getTextContent()获得节点值。**

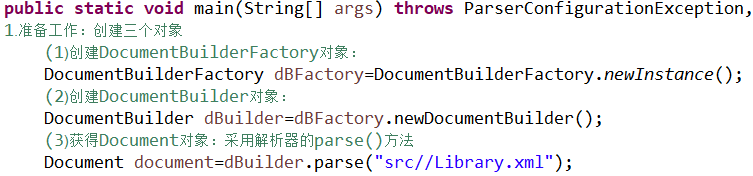
注意这里不能直接调用Node的getNodeValue()获得节点值。因为这里的节点类型是元素节点类型，对于元素节点，采用getNodeValue()方法只能得到null值。

**方法3：不对Node进行强转。直接调用Node的getNodeName()获得节点名。调用Node类的getFirstChild()方法获得子节点，然后再调用getNodeValue()方法获得节点值。**

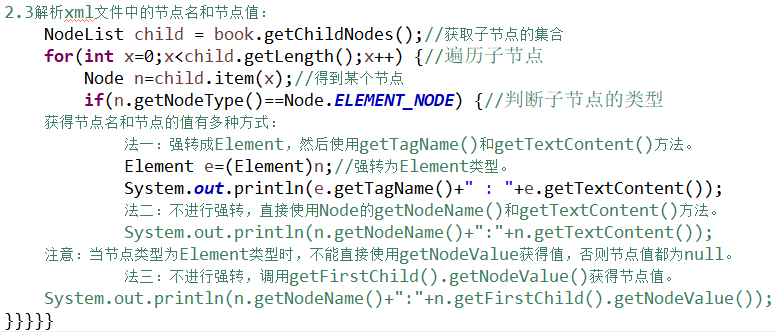
这是因为标签之间(起始标签和结束标签称为元素节点)的文本被看作是该标签的字节点，因此该Node的getFirstChild()获取的子节点是文本节点，而文本节点的getNodeValue()得到的是节点内容。注意，若该元素节点Node还存在其它子节点和子节点的值，如<name><a>小明</a>男性</name>，调用getFirstChild(). getNodeValue()获得的值还是null，因为获取的子节点仍然是元素节点类型。若使用的是getTextContent()，则获取的结果是“<a>小明</a>男性”，标签<a>也会被当成文本内容。

#### ⑶实例

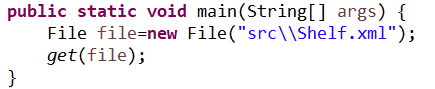
##### ①通过标签名获得节点集合且多个属性实例1(重点)







##### ②通过标签名获得节点集合且单个属性实例2(重点)：

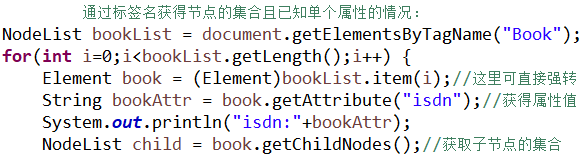




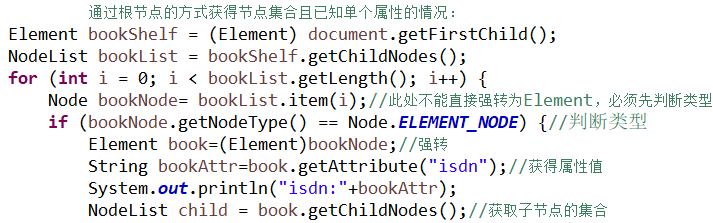
##### ③通过根节点获得集合并遍历且单个属性实例3(重点)：

通过根节点的方式获得节点集合并且已知单个属性的情况，和通过标签名获得节点的集合并且已知单个属性的情况，这两者在获得节点属性的代码上稍微有点区别。

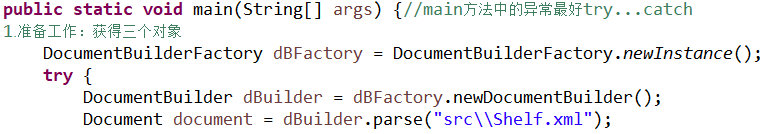
**通过标签名获得节点的集合且已知单个属性的情况：**



**通过根节点的方式获得节点集合且已知单个属性的情况：**



根节点中不能直接强转的原因：采用getFirstChild().getChildNodes()获得的节点集合中包含了文本节点类型。而文本节点类型不能强转成元素节点类型，否则运行时会抛出异常。因此不能直接进行强转，而要先进行类型判断之后，若为元素节点类型，则再进行强转为Element类型，然后再用Element的getAttribute()方法获得单个指定属性的属性值。而标签名中可以直接进行强转，且能正常运行。





## 二、DOM4J

DOM4J：Document Object Model for Java

### 1.DOM4J生成xml文件并添加数据

DOM4J中需要将额外的jar包通过Build Path的方式导入到项目中。

DOM4J中**导入的包**为：**import** org.dom4j.\*；

#### ⑴常用的方法

**DocumentHelper.createDocument()**：返回值为Document类型。在DOM4J中采用这种方法创建Document对象。

Branch**.addElement(String args)**：返回值为String类型，创建节点。

Document**.setRootElement(Element args)**：无返回值，将节点设置为根节点，参数为Element类型。

Element**.addAttribute(String args,String args2)**：返回值为Element，给节点添加属性，第一个参数为属性名，第二个参数为属性值。

Node**.setText(String args)**：无返回值，给节点设置内容。

FileWriter fw=**new** **FileWriter(File f)**:参数为要写入的文件。

**OutputFormat.createPrettyPrint()**：返回值为OutputFormat，即用来设置输出格式的。

OutputFormat**.setEncoding(String Encoding)**：无返回值，设置编码格式。

XMLWriter xw=**new** **XMLWriter(Writer args)**:括号内的参数为Writer类型，即将一个Writer类型的流以XMLWriter的形式写入。括号内的参数也可以为(Writer args,OutputFormat format)形式，第二个参数为输出的格式。

#### ⑵步骤

**1.创建document内存树：**

**采用DocumentHelper.createDocument()方法创建Document**

**创建节点：addElement()**

**将节点设置为根节点：setRootElement()**

**采用循环的方式向根节点中添加节点**

**添加节点**

**给添加的节点设置属性：addAttribute()**

**给节点添加子节点：**

**给添加的子节点设置内容：setText()**

**2.用流的方式将document对象写到指定的xml文件中**

**创建一个文件写入流：new FileWriter()**

**美化输出格式：OutputFormat.createPrettyPrint()**

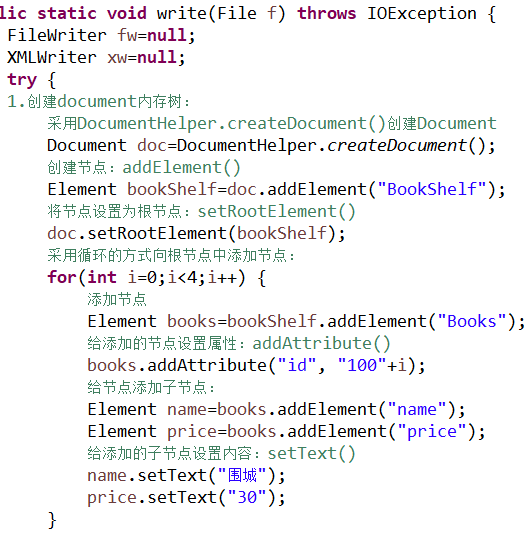
**设置编码格式**

**创建一个XML写入流：new XMLWriter()**

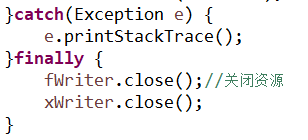
**将document对象写入xml文件中**

**3.关闭资源**

#### ⑶实例：







### 2.DOM4J解析xml文件获取数据

#### ⑴常用的方法

SAXReader sr=**new** SAXReader()：创建读取器。

SAXReader**.read(File f)**：返回值为Document类型。

Document**.getRootElement()**：返回值为Element，获得Document树的根节点。

Element**.Elements()**：返回值为List类型，返回的是列表形式的节点。

Element**.elementIterator()**：返回值为一个迭代器。这个迭代器可用来迭代Element节点。

Element**.attributeValue(String s)**：参数为String类型的属性名，该方法返回的是指定属性名的属性值。返回类型为String类型。

Element**.attributes()**：返回值为List类型，这个集合中存放的类型都是Attribute，即属性。

Attribute**.getValue()**：获得属性的值，返回值为String类型。

Node**.getName()**：返回值为String类型，得到节点的名称。

Element**.getText()**：返回值为String类型，得到节点的内容。

Element**.getStringValue()**：返回值String类型，获得节点的内容。

#### ⑵步骤

**DOM4J只能从根节点读取：**

**1.创建读取器：new SAXReader()**

SAXReader sReader=**new** SAXReader();

**2.读取文件,并获得其Document对象树：read()**

Document document=sReader.read("src\\School.xml");

**3.获得树的根节点：getRootElement()**

Element students=document.getRootElement();

**4.得到根节点下的节点集合：**

**法一：Element.elements()方法获得节点集合。**

List<Element> animalList = animals.elements();

**法二： Element.elementIterator()方法获得迭代器。**

Iterator it = students.elementIterator();

**5.遍历节点集合或者迭代器：**

**若是遍历迭代器，则要将获得的节点强转为Element类型**

Element student=(Element)it.next();

**6.获得节点的属性名和属性值：**

**情况1：单个属性且已知属性名，Element.attributeValue(String s)获得已知属性的值。**

System.***out***.println("id:"+e.attributeValue("id"));

**情况2：多个属性且未知属性名，通过Element.attributes()获得属性的集合，然后遍历该集合。通过getName()、getValue()获得属性名和属性值。**

List<Attribute> studentAttr = student.attributes();

System.***out***.println( attr.getName()+":"+attr.getValue());

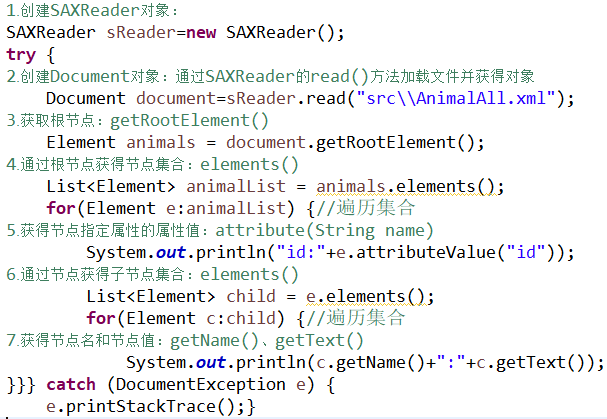
**7.获得节点的子节点集合：和步骤4一样，同样的两种方法。**

**8.遍历节点集合或者迭代器：通过getName()获得节点名，getText()或者getStringValue()或的节点值。**

System.***out***.println (child.getName()+":"+child.getStringValue());或者System.***out***.println(c.getName()+":"+c.getText());

#### ⑶实例：

**例1：for循环遍历节点集合且只有一个属性和已知属性名**



**例2：通过迭代器的方式遍历节点集合且多个属性**



## 三、DOM和DOM4J的总结

### 1.生成xml文件

⑴生成Document：

DOM：DocumentBuilderFactory.newInstance().DocumentBuilder().newDocument()

DOM4J：DocumentHelper.createDocument()

⑵创建结点：

DOM：createElement()

DOM4J：addElement()

⑶添加子结点：

DOM：appendChild()

DOM4J：addElement()，设置根结点setRootElement()

⑷给结点设置属性：

DOM：setAttribute()

DOM4J：addAttribute()

⑸给结点设置内容：

DOM：setTextContent()

DOM4J：setText()

⑹将Document内存树写入文件中

DOM：Transformer、new DOMSource()、new StreamResult()

DOM4J：new FileWriter()、new XMLWriter()

### 2.读取xml文件

⑴读取器：

DOM：newDocumentBuilder()(解析器)

DOM4J：SAXReader()

⑵读取方法：

DOM：parse()(解析)

DOM4J：read()

⑶获得根结点：

DOM：getFirstChild()

DOM4J：getRootElement()

⑷获得结点集：

DOM：getChildNodes()(子结点集)

DOM4J：Element.Elements()

⑸得到结点内容：

DOM：getTagName()、getTextContent()

DOM4J：getName()、getText()

⑹获得属性：

DOM：getAttribute()

DOM4J：Element.attributeValue()

## 四、SAX

SAX：Simple Api for Xml

有空再学习

## 五、JDOM

JDOM：Java-based Document Object Model

有空再学习

## 六、总结

**DOM解析会将整个xml文件加载到内存中，然后再逐个解析。SAX解析是通过Handler处理类逐个依次解析每个节点**。

**基础方法**：DOM(是与平台无关的官方解析方式)、SAX(是基于事件驱动的解析方式)。

**扩展方法**：JDOM、DOM4J。这两种方法是在基础的方法上扩展出来的，只有在java中能够使用的解析方法。

### 1.四种方式的优缺点

**⑴DOM的优缺点**

**优点**：形成了树结构，直观好理解，代码更易编写；解析过程中树结构保留在内存中，方便修改。访问是双向的，可在任何时候在树中上下导航，获取和操作任意部分的数据。

**缺点**：当xml文件较大时，对内存耗费比较大，容易影响解析性能并造成内存溢出。

**⑵SAX的优缺点**

**优点**：采用事件驱动模式，对内存耗费比较小，效率和性能较高；能够即时解析xml文档也能即时停止解析。

只在读取数据时检查数据，而不需要保存在内存中。分析能立即开始，而不需要等待所有的数据都被处理完。可在某个条件得到满足时停止解析，而不需要解析整个文档。

**缺点**：不易编码；很难同时访问同一个xml文件中的多处不同数据。

**⑶JDOM的优缺点**

**优点**：仅使用具体类而不使用接口，在某些方面简化了API；API中大量使用了Collections类，方便了java开发人员。

**缺点**：没有较好的灵活性；性能较差。

**⑷DOM4J的优缺点**

JDOM的一种智能分支，它合并了许多超出基本xml文档表示的功能。

DOM4J使用接口和抽象类的基本方法，是一个优秀的Java XML API。具有性能优异、灵活性好、功能强大和极端易使用的特点。

**DOM4J在灵活性和对复杂xml的支持上都要强于DOM**，在xml文件较大时，DOM4J更好用。**在多数情况下都用DOM4J**。

**优点**：大量使用了java集合类，方便java开发人员，同时提供了一些提高性能的替代方法；支持XPath；性能优异、灵活性好、功能强大等特点。

**缺点：**大量使用了接口，API较复杂。

### 2.四种方式的比较分析

⑴DOM4J的性能最好，若不考虑移植性，就采用DOM4J的方式。

⑵JDOM和DOM在性能测试时表现不佳，在测试较大xml文档时，容易内存溢出(如10M的文档)，但可移植。DOM的应用也挺广泛，广泛应用于多种编程语言。如JavaScript中使用DOM。

⑶SAX的性能测试表现较好，若需要即时处理数据而不需要保存数据则可以考虑使用SAX。