1. XML解析方式（原理）：
   1. DOM 解析树
   2. SAX 流事件

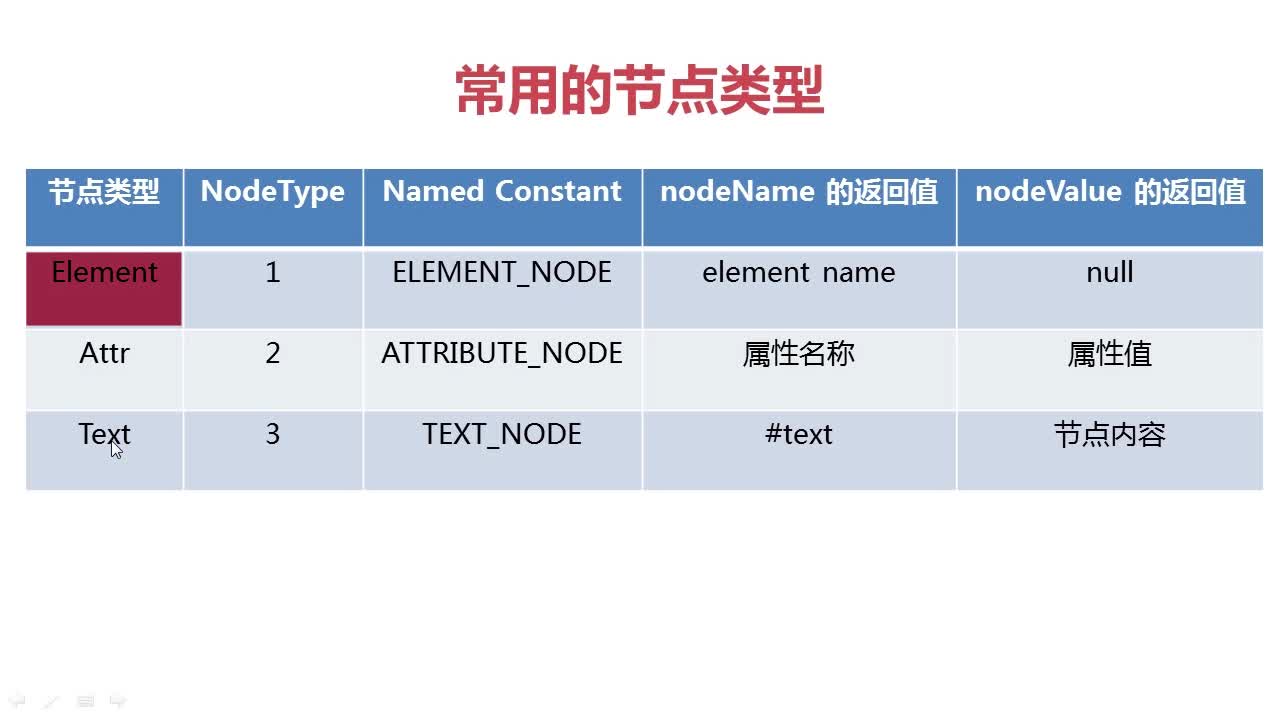
DOM解析对应主流工具

* + 1. DOM（官方）
    2. DOM4J （非官方 主流 三大框架使用DOM4J解析配置文件）
    3. JDOM （非官方）

SAX解析对应主流工具

SAX （官方 主流）

1. 解析XML文档
   1. 要处理XML文档就要先解析(parse)它，
   2. 解析器程序：读入一个文件，确认这个文件具有正确的格式，然后分解成各种元素，使程序员能够访问这些元素
      1. 文档对象模型（Document Object Model，DOM）解析器，是**树型**解析器（tree parser），将读入的XML文档转换成**树结构**
      2. XML简单API（Simple API for XML，SAX）解析器，是**流机制**解析器（streaming parser），在读入XML文档时生成相应的**事件**
   3. 如果只是对于**某些元素**感兴趣，而不关心他们的上下文，那么在这些情况下你应该考虑使用流机制解析器
2. 节点Node
   1. 一个XML文档中只有一个根节点
   2. 没有父节点的元素节点
   3. 元素节点
   4. 根节点和根节点中的所有标签都是元素节点
   5. 属性节点
   6. <student id=“001”></student> id就是属性节点
   7. 文本节点
   8. <name>zhangsan</name> zhangsan就是文本节点
   9. 注释节点
   10. <!-- --> xml文档中的注释信息



Element类型：带标签的<a>b</a>，**节点名是a，节点值都是null**，b是“节点值”，也是子节点（此时b是Text类型的节点）

Attr类型：**节点值是属性名，节点值是属性值**

Text类型：在标签之外的都是文本节点，**节点名都是#text，节点值是文本**

因此，我们认为的Element类型的节点的“节点值”，实际上是Element类型节点的Text类型的子节点——指夹在标签中间的内容，不能通过getNodeValue()获取，要通过：

1. getFirstChild().getNodeValue() 获得第一个子节点的节点值
2. getTextContent() 获得所有**Text子节点**的节点值，也会递归孙子节点获得曾孙节点的节点值
   1. 例如<a><b>bb</b>哈哈哈</a>将得到：bb哈哈哈
   2. 而用1.将解析b这个Element节点，将得到null
3. DOM官方解析示例：
   1. 首先要明白XML是被看成文档（Document）的，所以我们需要一个DocumentBuilder对象来读取XML，生成对应的Document对象
      1. /\*用Builder工厂生成Builder\*/
      2. DocumentBuilderFactory factory = DocumentDuilderFactory.newInstance();
      3. DocumentBuilder builder = factory.newDocumentBuilder();
      4. /\*用builder读取XML文档生成Document对象\*/
      5. File f = new File(“…”);
      6. Document doc = builder.parse(f); //也可以读URL或者输入流
   2. 启动对Document的分析，**调用getDocumentElement，获得root元素**
      1. Element root = doc.getDocumentElement();
   3. 获得root节点子节点的列表，
      1. 注意，子节点不仅有Element元素类型还有**Text文本类型（不在标签里的，连换行回车的空白也算**）
      2. NodeList children = root.getChildNodes();
      3. For(int i=0; i<children.getLength(); i++){

Node child = children.item(i);

……

* + 1. }
    2. 如果只希望得到元素子节点（Element），那么可以忽略空白字符（Text）：
    3. NodeList children = root.getChildNodes();
    4. For(int i=0; i<children.getLength(); i++){

Node child = children.item(i);

if(child instanceof Element){

Element childElement = (Element)child;

…

}

* + 1. }
  1. 如果文档有DTD，那么解析器知道哪些是没有文本节点的子元素，而且会自动剔除掉空白字符！DTD很好用
  2. 如<a>123</a><b>456</b>，当我们想知道标签夹着的信息时，**既然Text是这些元素子节点唯一的子节点**，就可以用getFirstChild方法而不必对着这些元素子节点的NodeList一顿遍历，而只需要之后再对Text节点getData得到字符串即可
     1. For(int i=0; i<children.getLength(); i++){
        1. Node child = children.item(i);
        2. If(child instanceof Element){
           1. Element childElement = (Element)child;
           2. Text textNode = (Text)childElement.getFirstChild();
           3. String text = textNode.getData().trim(); //去掉多余空格和换行符
           4. if(childElement.getTagName().equals(“name”)) //name标签名

name = text;

* + - * 1. else if(childElement.getTagName().equals(“size”))

size = Integer.parseInt(text);

}

}

对元素节点的文本子节点getData后再trim，可以优化这种情况：

<size>

36

</size>

**此时文本子节点中含有换行和空格，调用trim可以删掉前后的空格**

* 1. 枚举元素子节点的属性<size hel=”pt”>36</size>
     1. 一个标签可以有多个属性，每个属性都有它自己的名称和取值，例如：

<input name=“text”>

属性值一定要用双引号（"）或单引号（'）引起来

定义属性必须遵循与标签相同的命名规范

多学一招：在XML技术中，标签属性所代表的信息，也可以被改成用子元素的形式来描述，例如：

<input>

**<name>text</name>**

</input>

**注意：此时就不能使用getAttributes了**

* + 1. 文档对象调用getAttributes，返回一个NameNodeMap对象，其中包含了描述属性的Node对象（键值对），遍历该Map得Node，调用getNodeName和getNodeValue获得属性名和属性值
    2. 或者知道属性名，直接获取属性值
       1. String unit = element.getAttrbute(“unit”); //unit是属性名，将属性值赋给String变量unit

或者

外部

<!DOCTYPE configuration PUBLIC “http://myserver.com/config.dtd”>