一、XML是什么？作用是什么？  
xml是用来传输和存储数据的，本身对数据没有任何操作。在这里要区别一下html，html是用来显示数据的。xml的焦点是数据内容，html的焦点是数据外观。

下面是xml的定义：

•XML 指可扩展标记语言（EXtensible Markup Language）

•XML 是一种标记语言，很类似 HTML

•XML 的设计宗旨是传输数据，而非显示数据

•XML 标签没有被预定义。您需要自行定义标签。

•XML 被设计为具有自我描述性。

•XML 是 W3C 的推荐标准

xml没什么特别的。它仅仅是纯文本而已。有能力处理纯文本的软件都可以处理xml。不过，能够读懂xml的应用程序可以有针对性地处理xml的标签。标签的功能性意义依赖于应用程序的特征。xml允许创作者定义自己的标签和文档结构。  
2、用途

XML应用于web开发的许多方面，常用于简化数据的存储和共享。

1、xml将数据从HTML中分离出来。这样只需要javascript就读取一个外部的xml文件。用来更新HTML，这样HTML就可以专注的显示外观。

2、xml简化了数据共享。在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据。xml数据以纯文本格式进行存储，因此提供了一种独立于软件和硬件的数据存储方法。

3、xml简化数据传输。通过xml，可以在不兼容的系统之间轻松地交换数据。对开发人员来说，其中一项最费时的挑战一直是在因特网上的不兼容系统之间交换数据。由于可以通过各种不兼容的应用程序来读取数据，以xml交换数据降低了这种复杂性。

4、xml简化平台的变更。升级到新的系统（硬件或软件平台），总是非常费时的。必须转换大量的数据，不兼容的数据经常丢失。xml数据以文本格式存储。这使得xml在不损失数据的情况下，更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

5、xml使您的数据更有用。由于xml独立于硬件、软件以及应用程序，xml使您的数据更可用，也更有用。不同的应用程序都能够访问您的数据，不仅仅在HTML页中，也可以从xml数据源中进行访问。

总之xml的跨平台性是非常好用的。

3、XML的树形结构

1. **<?xml** version="1.0" encoding="UTF-8"**?>**
2. **<note>**
3. **<to>**George**</to>**
4. **<from>**John**</from>**
5. **<heading>**Reminder**</heading>**
6. **<body>**Don't forget the meeting!**</body>**
7. **</note>**

note{to,form,heading,body}

第一行是XML声明。它定义XML的版本（1.0）和使用的编码（ISO-8859-1=Latin-1/西欧字符集）。

下一行描述文档的根元素<note>

根元素开始标签和结束标签里面包含的是4个子元素

xml文档形成一种树结构。

xml文档必须包含根元素。该元素是所有其他元素的父元素。

xml文档中的元素形成了一棵文档树。这棵树从根部开始，并扩展到树的最底端。

所有元素均可拥有子元素。

1. **<root>**
2. **<child>**
3. **<subchild>**...**</subchild>**
4. **</child>**
5. **</root>**

二、XML的基本语法  
1、文档声明  
作用：用于标识该文档是一个XML文档。  
注意事项：声明必须出现在文档的第一行（之前连空行都不能有，也不能有任何的注释）  
  
  
最简单的XML声明：<?xml version="1.0"?>  
声明中的encoding属性：说明解析当前XML文档时所使用的编码。默认是UTF-8  
声明中的standalone属性：说明XML文档是否是独立的。（了解）  
  
  
  
  
2、元素  
结束标签不能省略  
一个XML文档必须且只能有一个根标签  
XML文档中不会忽略回车和换行及空格  
标签的命名规范：元素(标签)的名称可以包含字母、数字、减号、下划线和英文句点。严格区分大小写。  
  
  
3、元素的属性  
元素的属性取值一定要用引号引起来（单引号或双引号）  
  
  
4、注释  
与HTML中的注释完全一致：<!--这是注释-->  
注释不能嵌套  
  
  
  
  
5、CDATA区  
CDATA是Character Data的缩写。  
作用：CDATA区中的东东都是文本。  
语法：  
<![CDATA[  
  
文本内容 > < &  
  
]]>  
6、特殊字符  
& &amp;  
< &lt;   (less than)  
> &gt:  (great than)  
" &quot;  
' &apos;  
  
  
7、处理指令(PI:Processing Instruction)  
处理指令，简称PI(Processing Instruction)。  
作用：用来指挥软件如何解析XML文档。  
语法：必须以“<?”作为开头，以“?>”作为结尾。  
  
  
三、XML的约束  
1、格式良好的XML文档：符合XML语法的。  
2、有效的XML文档：遵循约束规范的。  
格式良好的不一定是有效的，但有效的必定格式良好。  
  
  
  
  
四、DTD的基本语法（看懂即可）

**拥有正确语法的 XML 是形式良好的 XML。**

**通过某种 DTD 进行了验证的 XML 是合法的 XML。**

形式良好的 XML 文档

一个"形式良好"的 XML 文档拥有正确的语法。

一个"形式良好"的 XML 文档会遵守XML 语法规则：

* XML 文档必须有根元素
* XML 文档必须有关闭标签
* XML 标签对大小写敏感
* XML 元素必须被正确的嵌套
* XML 属性必须加引号

1、DTD：Document Type Definition  
2、作用：约束XML的书写规范。  
3、DTD文件保存到磁盘时，必须使用UTF-8编码  
  
  
4、如何引入外部的DTD文档来约束当前的XML文档  
DTD文件在本地：<!DOCTYPE 根元素名称 SYSTEM "DTD文件的路径">  
DTD文件在网络上：<!DOCTYPE 根元素名称 PUBLIC "DTD名称" "DTD的路径URL">  
  
  
5、DTD的语法细节

5.1定义元素  
语法：<!ELEMENT 元素名称 使用规则>  
使用规则：  
(#PCDATA):指示元素的主体内容只能是普通的文本.(Parsed Character Data)  
EMPTY:指示元素的不能有主体内容。  
ANY:用于指示元素的主体内容为任意类型  
(子元素)：指示元素中包含的子元素  
如果子元素用逗号分开，说明必须按照声明顺序去编写XML文档  
如果子元素用“|”分开，说明任选其一。  
用+、\*、？来表示元素出现的次数

5.2定义元素的属性(attribute)  
语法：<!ATTLIST 哪个元素的属性  
属性名1 属性值类型 设置说明  
属性名2 属性值类型 设置说明>  
属性值类型：  
CDATA：说明该属性的取值为一个普通文本  
ENUMERATED (DTD没有此关键字)：  
语法：<!ATTLIST 元素名称 (值1|值2) "值1">  
ID:属性的取值不能重复  
设置说明：  
#REQUIRED：表示该属性必须出现  
#IMPLIED：属性可有可无  
#FIXED:表示属性的取值为一个固定值 语法：#FIXED "固定值"  
直接值：表示属性的取值为该默认值

5.2定义实体  
关键字ENTITY  
实体的定义分为引用实体和参数实体  
引用实体：  
作用：在DTD中定义，在XML中使用  
语法：<!ENTITY 实体名称 "实体内容">  
在XML中使用：&实体名称;  
参数实体：  
作用：在DTD中定义，在DTD中使用  
语法：<!ENTITY % 实体名称 "实体内容">  
在DTD中使用：%实体名称;  
  
  
五、XML解析方式概述  
1、常用XML的解析方式：DOM和SAX  
DOM:Document Object Model是W3C推荐使用的解析方式

DOM是用与平台和语言无关的方式表示XML文档的官方W3C标准。DOM是以层次结构组织的节点或信息片断的集合。这个层次结构允许开发人员在树中寻找特定信息。分析该结构通常需要加载整个文档和构造层次结构，然后才能做任何工作。由于它是基于信息层次的，因而DOM被认为是基于树或基于对象的。DOM以及广义的基于树的处理具有几个优点。首先，由于树在内存中是持久的，因此可以修改它以便应用程序能对数据和结构作出更改。它还可以在任何时候在树中上下导航，而不是像SAX那样是一次性的处理。DOM使用起来也要简单得多。

为 XML 文档的已解析版本定义了一组接口。解析器读入整个文档，然后构建一个驻留内存的树结构，然后代码就可以使用 DOM 接口来操作这个树结构。

优点：整个文档树在内存中，便于操作；支持删除、修改、重新排列等多种功能；

缺点：将整个文档调入内存（包括无用的节点），浪费时间和空间；使用场合：一旦解析了文档还需多次访问这些数据；硬件资源充足（内存、CPU）。

新增：

DocumentBuilderFactory

DocumentBuilder

Document

Element 🡪 document.appendChild ->setAttribute->Node

TransformerFactory  ->Transformer ->setOutputProperty(OutputKeys.ENCODING, "utf-8")

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

->StreamResult ->transform(xmlSource , StreamResult)

解析：

DocumentBuilderFactory –> DocumentBuilder -> Document -> DocumentBuilder  parse(Document)-> NodeList ->遍历解析-> node.getNodeType() == Node.ELEMENT\_NODE or Node.TEXT\_NODE-> NamedNodeMap map = node.getAttributes() ;重新遍历解析子节点

SAX:Simple API for XML。非官方标准。

SAX处理的优点非常类似于流媒体的优点。分析能够立即开始，而不是等待所有的数据被处理。而且，由于应用程序只是在读取数据时检查数据，因此不需要将数据存储在内存中。这对于大型文档来说是个巨大的优点。事实上，应用程序甚至不必解析整个文档；它可以在某个条件得到满足时停止解析。一般来说，SAX还比它的替代者DOM快许多。

选择DOM还是选择SAX？ 对于需要自己编写代码来处理XML文档的开发人员来说， 选择DOM还是SAX解析模型是一个非常重要的设计决策。

DOM采用建立树形结构的方式访问XML文档，而SAX采用的事件模型。

DOM解析器把XML文档转化为一个包含其内容的树，并可以对树进行遍历。用DOM解析模型的优点是编程容易，开发人员只需要调用建树的指令，然后利用

navigation

APIs访问所需的树节点来完成任务。可以很容易的添加和修改树中的元素。然而由于使用DOM解析器的时候需要处理整个XML文档，所以对性能和内存的要求比较高，尤其是遇到很大的XML文件的时候。由于它的遍历能力，DOM解析器常用于XML文档需要频繁的改变的服务中。

SAX解析器采用了基于事件的模型，它在解析XML文档的时候可以触发一系列的事件，当发现给定的tag的时候，它可以激活一个回调方法，告诉该方法制定的标签已经找到。SAX对内存的要求通常会比较低，因为它让开发人员自己来决定所要处理的tag.特别是当开发人员只需要处理文档中所包含的部分数据时，SAX这种扩展能力得到了更好的体现。但用SAX解析器的时候编码工作会比较困难，而且很难同时访问同一个文档中的多处不同数据。

优点：不用事先调入整个文档，占用资源少；SAX解析器代码比DOM解析器代码小，适于Applet，下载

缺点：不是持久的；事件过后，若没保存数据，那么数据就丢了；无状态性；从事件中只能得到文本，但不知该文本属于哪个元素；使用场合：Applet;只需XML文档的少量内容，很少回头访问；机器内存少；

解析：

SAXParserFactory –> SAXParser ->parse(**new** FileInputStream(fileName), **自定义解析器**)

->自定义解析器 **extends** DefaultHandler ...->startElememnt(){节点名称...遍历属性}->characters(){节点文本信息}

2、常用解析开发包：  
JDom

JDOM的目的是成为Java特定文档模型，它简化与XML的交互并且比使用DOM实现更快。由于是第一个Java特定模型，JDOM一直得到大力推广和促进。正在考虑通过“Java规范请求JSR-102”将它最终用作“Java标准扩展”。从2000年初就已经开始了JDOM开发。

JDOM与DOM主要有两方面不同。首先，JDOM仅使用具体类而不使用接口。这在某些方面简化了API，但是也限制了灵活性。第二，API大量使用了Collections类，简化了那些已经熟悉这些类的Java开发者的使用。

JDOM文档声明其目的是“使用20%（或更少）的精力解决80%（或更多）Java/XML问题”（根据学习曲线假定为20%）。JDOM对于大多数

Java/XML应用程序来说当然是有用的，并且大多数开发者发现API比DOM容易理解得多。JDOM还包括对程序行为的相当广泛检查以防止用户做任何在XML中无意义的事。然而，它仍需要您充分理解XML以便做一些超出基本的工作（或者甚至理解某些情况下的错误）。这也许是比学习DOM或JDOM接口都更有意义的工作。

JDOM自身不包含解析器。它通常使用SAX2解析器来解析和验证输入XML文档（尽管它还可以将以前构造的DOM表示作为输入）。它包含一些转换器以将JDOM表示输出成SAX2事件流、DOM模型或XML文本文档。JDOM是在Apache许可证变体下发布的开放源码。

为减少DOM、SAX的编码量，出现了JDOM；

优点：20-80原则，极大减少了代码量。使用场合：要实现的功能简单，如解析、创建等，但在底层，JDOM还是使用SAX（最常用）、DOM、Xanan文档。

创建：

Document ->Element ->{属性(setAttribute) 子节点 (addContent) }-> Format.getCompactFormat()->设置setEncoding、setIndent-> XMLOutputter (Format)->

->output(Document ,  **new** FileOutputStream(fileName))

解析：

SAXBuilder –> build(**new** FileInputStream(fileName))->Document-> getRootElement()->遍历解析(Element )->{ isRootElement or 节点 -属性-重新遍历}

Dom4J  
虽然DOM4J代表了完全独立的开发结果，但最初，它是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，包括集成的XPath支持、

XML Schema支持以及用于大文档或流化文档的基于事件的处理。它还提供了构建文档表示的选项，它通过DOM4J

API和标准DOM接口具有并行访问功能。从2000下半年开始，它就一直处于开发之中。

为支持所有这些功能，DOM4J使用接口和抽象基本类方法。DOM4J大量使用了API中的Collections类，但是在许多情况下，它还提供一些替代方法以允许更好的性能或更直接的编码方法。直接好处是，虽然DOM4J付出了更复杂的API的代价，但是它提供了比JDOM大得多的灵活性。

在添加灵活性、XPath集成和对大文档处理的目标时，DOM4J的目标与JDOM是一样的：针对Java开发者的易用性和直观操作。它还致力于成为比

JDOM更完整的解决方案，实现在本质上处理所有Java/XML问题的目标。在完成该目标时，它比JDOM更少强调防止不正确的应用程序行为。

DOM4J是一个非常非常优秀的Java XML

API，具有性能优异、功能强大和极端易用使用的特点，同时它也是一个开放源代码的软件。如今你可以看到越来越多的Java软件都在使用DOM4J来读写

XML，特别值得一提的是连Sun的JAXM也在用DOM4J.  
  
  
创建：

DocumentHelper创建🡪Document->添加addElement(str)->addAttribute->setText(str)文本值

OutputFormat->setEncoding->setIndent->setNewLines

XMLWriter->new XMLWriter(outputStream, OutputFormat)-> XMLWriter writer(Document)->close()

解析：

SAXReader->read(InputStream)->Document->getRootElement()->遍历解析子元素

🡪解析属性 attributeIterator()🡪嵌套解析子节点