# 1.XML

## 1.1.xml是什么

xml：（Extensible Markup Language）可扩展标记语言。

由w3c组织提供，2000年开发出1.0版本

xml是一种数据存储格式，本质上就是一段字符串

## 1.2.xml是如何保存数据的

场景：保存中国的省份和城市信息

在XML中, 允许用户自己定义标签, 标签分为开始标签和结束标签, 开始标签和结束标签之间又可以嵌套其他的标签.

正是利用标签来保存数据, 利用标签之间的嵌套关系来保存数据之间的层级关系.

## 1.3.xml的应用场景

(1)传输数据

由于xml本质上是一段字符串, 具有跨平台的特性, 因此可以在不同系统之间进行数据交换。

比如天气数据，xml接口：<http://flash.weather.com.cn/wmaps/xml/china.xml>

(2)用作配置文件

由于xml可以保存有结构的数据, 因此xml常被用来用作应用程序的配置文件。

## 1.4.xml文件、xml校验

我们可以使用浏览器对XML进行校验。

## 1.5.xml语法

### 1.5.1．文档声明

可以用来声明xml文档的基本属性信息, 解析器会根据文档声明来决定如何解析这个xml文档.

注意:

一个xml文档有且仅有一个文档声明.(规范要求)

文档声明只能放在xml文档的第一行, 并且前面不能有任何内容

如果一个xml文档没有文档声明, 该文档是一个格式不良好的xml文档.

<?xml version=”1.0” encoding=”utf-8”?>

其中version用来指定xml文档所遵循的xml版本, 目前就是1.0， encoding属性用来通知解析器使用什么编码来解析当前xml文档. 注意, encoding属性指定的编码和xml文档保存时的编码必须得一致!!!

### 1.5.2．元素

一个标签就是一个元素, 如: <中国></中国>

标签体：标签分为开始标签和结束标签, 开始标签和结束标签之间的文本。

如：<a>xxxxx</a>

自闭标签：如果一个元素不包含标签体也不包含其他元素，那么可以将开始标签和结束标签合并

如：<a></a> ---> <a/>

根标签：一个xml文档有且仅有一个根标签

标签可以包含子标签，但一定要合理的嵌套, 不能出现交叉嵌套.

比如:

<a>welcome to <b>www.tarena.com.cn</a></b>

<a>welcome to www.tarena.com.cn</b></a>

<a>welcome to www.tarena.com.cn<b/></a>

<a>welcome to www.tarena<b/>.com.cn</a>

XML元素的命名规范:

区分大小写, 如: <a>和<A>是两个不同的标签

不能以数字或标点符号开头, 如: <123a>

不能以xml（或Xml、或XML）开头

不能包含空格

不能包含冒号

### 1.5.3．属性

一个标签可以有多个属性，多个属性之间用空格隔开，属性值一定要用双引号（"）或单引号（'）引起来，例如：

<中国 pyname=”zhongguo” gdp=”80万亿”>

属性和元素具有相同的命名规范(参考上面)

### 1.5.4.注释

格式: <!-- 一段注释 -->

注意:

注释不能放在文档声明的前面

注释不能交叉嵌套, 如:

<!-- <!-- 注释内容 注释内容--> 注释内容 --> 这是不合理的注释

### 1.5.5.转义字符

用其他符号来代替特殊符号

< &lt;

> &gt;

“ &quot;

‘ &apos;

& &amp;

**5.6.CDATA区**

它的全称为character data,以"<![CDATA[ "开始，以" ]]>" 结束，在两者之间嵌入不想被解析程序解析的原始数据，解析器不对CDATA区中的内容进行解析，而是将这些数据原封不动地交给下游程序处理。

<![CDATA[

....//可以输入任意字符（除了]]>），不能嵌套。

]]>

## 1.6.xml约束

在xml技术里，可以编写一个文档来约束一个xml文档的写法，这称之为XML约束。

分为两种：DTD和Schema W3C组织提供

## 1.7.xml解析

xml两种解析方式：DOM解析和SAX解析（参见DOM解析图和SAX解析图）

解析开发包：jaxp(sun) jdom dom4j

Dom4j解析：

dom4j是一个开源的XML解析包，具有性能优异、功能强大和极其易使用的特点。

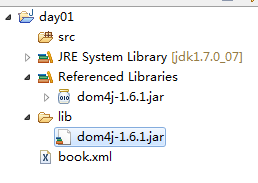
**案例：操作元素**

**案例一：查询第一本书的书名，并输出到控制台**

1.创建xml文档：在myEclipse中创建day01项目，在项目中创建book.xml，将dom4j解析用例中的内容拷贝到book.xml中

|  |
| --- |
| <?xml version="1.0" encoding="utf-8" ?>  <书架>  <书>  <书名>数据结构</书名>  <作者>严蔚敏</作者>  <售价>29.00元</售价>  </书>  <书>  <书名>高等数学</书名>  <作者>同济大学数学系</作者>  <售价>55.00元</售价>  </书>  </书架> |

2.导包：在项目中创建lib目录，将javaweb开发软件中的dom4j-1.6.1.jar复制到lib目录下，并build path。



3.创建demo1并加入如下代码

|  |
| --- |
| **package** cn.tedu;  **import** org.dom4j.Document;  **import** org.dom4j.DocumentException;  **import** org.dom4j.Element;  **import** org.dom4j.io.SAXReader;  **public** **class** Demo1 {  **public** **static** **void** main(String[] args) **throws** DocumentException {  //1.创建解析器  SAXReader reader = **new** SAXReader();  //2.利用解析器读取xml文件  Document dom = reader.read("book.xml");  //3.获取根节点  Element root = dom.getRootElement();  //4.获取第一本书节点  Element bookEle = root.element("书");  //5.获取第一个书名节点  Element bookNameEle = bookEle.element("书名");  //6.获取书名中的内容  String text = bookNameEle.getText();  System.*out*.println("第一本书的书名为："+text);  }  } |

**案例二：查询第二本书的售价，并输出到控制台**

|  |
| --- |
| **package** cn.tedu;  **import** java.util.List;  **import** org.dom4j.Document;  **import** org.dom4j.DocumentException;  **import** org.dom4j.Element;  **import** org.dom4j.io.SAXReader;  /\*\*  \* dom4j操作元素节点  \*/  **public** **class** Demo2 {  **public** **static** **void** main(String[] args) **throws** DocumentException {  //2.查询第二本书的售价，并输出到控制台  //1.创建解析器  SAXReader reader = **new** SAXReader();  //2.利用解析器读取xml文件  Document dom = reader.read("book.xml");  //3.获取根节点  Element root = dom.getRootElement();  //4.获取root下所有节点集合  List<Element> list = root.elements();  //5.获取第二本书节点  Element bookEle2 = list.get(1);  //6.获取第二本书的售价节点  Element price = bookEle2.element("售价");  //7.获取售价  String text = price.getText();  System.*out*.println("第二本书的售价为："+text);  }  } |

为了方便开发，将案例二中的代码抽取到一个方法中，可以通过快捷键alt+shift+m快速抽取。

发现获取document部分大量重复，抽取到一个工具类中，代码如下

|  |
| --- |
| /\*\*  \* 解析xml文件，获取document对象  \* **@param** xmlPath  \* **@return**  \*/  **public** **static** Document getDoc(String xmlPath){  **try** {  //1.创建解析器  SAXReader reader = **new** SAXReader();  //2.利用解析器读取xml文件  Document dom = reader.read(xmlPath);  **return** dom;  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException(e);  } |

**案例三：给第一本书添加一个特价节点（2种方式）**

|  |
| --- |
| //创建解析器并获取根节点  Document doc = XMLUtils.getDoc("book.xml");  Element root = doc.getRootElement();  //获取第一本书  Element bookEle = root.element("书");  /\*//方式一  //创建一个游离的节点  Element priceEle2 = DocumentHelper.createElement("特价");  priceEle2.setText("9.9元");  //将游离节点挂载到第一本书上  bookEle.add(priceEle2);\*/    //方式二  //在第一本书上添加特价子节点  Element priceEle2 = bookEle.addElement("特价");  priceEle2.setText("8.8元");    //将更新的document写入到xml中,并加入格式输出器  XMLWriter writer = **new** XMLWriter(  **new** FileOutputStream(**new** File("book.xml")),OutputFormat.*createPrettyPrint*());  writer.write(doc);  writer.close(); |

再次发现，输出到xml的代码也会大量重复，再次抽取到工具类中

|  |
| --- |
| /\*\*  \* 将更新的document写入到xml中  \* **@param** doc  \* **@param** xmlPath  \*/  **public** **static** **void** write2Xml(Document doc, String xmlPath){  **try** {  //将更新的document写入到xml中,并加入格式输出器  XMLWriter writer = **new** XMLWriter(  **new** FileOutputStream(**new** File(xmlPath)),OutputFormat.*createPrettyPrint*());  writer.write(doc);  writer.close();  } **catch** (Exception e) {  e.printStackTrace();  **throw** **new** RuntimeException(e);  }  } |

**案例四：给第二本书在作者节点前插入一个特价节点**

|  |
| --- |
| //创建解析器并获取根节点  Document doc = XMLUtils.getDoc("book.xml");  Element root = doc.getRootElement();  //获取root下所有节点集合  List<Element> list = root.elements();  //获取第二个元素  Element bookEle2 = list.get(1);  //创建一个游离节点  Element priceEle2 = DocumentHelper.*createElement*("特价");  priceEle2.setText("6.6元");  //将游离节点插入到指定位置  List list2 = bookEle2.elements();  list2.add(1, priceEle2);  //将更新的document写入到xml中  XMLUtils.*write2Xml*(doc, "book.xml");  System.*out*.println("执行完毕"); |

**案例五：删除第二本书的特价节点（2种方式）**

|  |
| --- |
| //创建解析器并获取根节点  Document doc = XMLUtils.*getRootElement*("book.xml");  Element root = doc.getDoc();  //获取root下所有节点集合  List<Element> list = root.elements();  //获取第二个本书  Element bookEle2 = list.get(1);  /\*//方法一  //获取特价节点  Element priceEle2 = bookEle2.element("特价");  //将特价节点从第二本书中删除  bookEle2.remove(priceEle2);\*/    //方法二  //获取第二本书所有元素  List list2 = bookEle2.elements();  list2.remove(1);  //将更新的document写入到xml中  XMLUtils.*write2Xml*(doc, "book.xml"); |

**案例六：更新第一本书的特价节点的内容为19.8元**

|  |
| --- |
| //创建解析器并获取根节点  Document doc = XMLUtils.getDoc("book.xml");  doc.getRootElement().element("书").element("特价").setText("19.8元");  //将更新的document写入到xml中  XMLUtils.*write2Xml*(doc, "book.xml"); |

**操作属性**

**案例七：给第一本书添加一个属性，如：出版社="清华大学出版社"(2种方式)**

|  |
| --- |
| //创建解析器，读取xml，获取根目录  Document doc = XMLUtils.*getDoc*("book.xml");  Element root = doc.getRootElement();  //获取第一本书  Element bookEle = root.element("书");    /\*//方法一  //创建一个游离的属性  Attribute attr = DocumentHelper.createAttribute(bookEle, "出版社", "清华大学出版社");  //将游离的属性挂载到第一本书上  bookEle.add(attr);\*/    //方法二  //在第一本书上直接添加属性  bookEle.addAttribute("版次","1.0");    //将更新的document写入到xml文档中  XMLUtils.*write2Xml*(doc, "book.xml"); |

**案例八：在控制台上打印输出第一本书的出版社属性的值,并更新属性的值为“人民出版社”(3种方式)**

|  |
| --- |
| //创建解析器，读取xml，获取根目录  Document doc = XMLUtils.*getDoc*("book.xml");  Element root = doc.getRootElement();  //获取第一本书  Element bookEle = root.element("书");    //方式一  //Attribute attr = bookEle.attribute(0);    //方式二  Attribute attr = bookEle.attribute("出版社");  attr.setValue("人民出版社");  System.*out*.println(attr.getName()+":"+attr.getValue());    //方式三  //String value = bookEle.attributeValue("出版社");  //System.out.println(value);    //将更新的document写入到xml文档中  XMLUtils.*write2Xml*(doc, "book.xml"); |

**案例九：删除第一本书的出版社属性(2种方式)**

|  |
| --- |
| //创建解析器，读取xml，获取根目录  Document doc = XMLUtils.*getDoc*("book.xml");  Element root = doc.getRootElement();  //获取第一本书  Element bookEle = root.element("书");    /\*//方式一  //获取第一本书的属性  Attribute attr = bookEle.attribute("出版社");  //将属性从第一本书中删除  bookEle.remove(attr);\*/    //方式二  //如果给一个正常值，为添加，如果给一个null，则为删除  bookEle.addAttribute("出版社",**null**);    //将更新的document写入到xml文档中  XMLUtils.*write2Xml*(doc, "book.xml"); |

## 1.8.作业练习

1.什么是B/S 结构,C/S 结构？

答：B/S为浏览器服务器模型，C/S为客户端服务器模型

B/S优点：无需下载客户端，只需浏览器即可访问，升级程序只需要升级服务端。

B/S缺点：页面展示能力较差，对网速依赖高。

C/S优点：页面展示能力强，对网速依赖较低。

C/S缺点：需要下载客户端，升级程序时，客户端也要升级。

2.什么是格式良好的 XML  
答：这个问题经常在电话面试中出现。一个格式良好的 XML 意味着该 XML 文档语法上是正确的，比如它有一个根元素，所有的开放标签合适地闭合，属性值必须加引号等等。如果一个 XML 不是格式良好的，那么它可能不能被各种 XML 解析器正确地处理和解析。

3.DOM 和 和 SAX  解析器有什么区别  
答：这又是一道常见面试题，使用 DOM 解析时，XML 文档以树形结构的形式加载到内存中，可以进行增删改查。而 SAX 是事件驱动的解析器，只能读，不能增删改。

dom优点：

1.可以对文档进行增删改

2.内存中保存了文档树，解析一次，即可重复使用

dom缺点：

1. 如果文件过大，有可能造成内存溢出
2. 需要把所有内容读完才能解析，效率较低

sax优点：

1. 逐行解析，无需将整个文档加载到内存，占用内存小，理论上多大的xml文档都能读
2. 解析时，遇到想要的内容可以随时停下来，效率高。

sax缺点：

1. 只能读，不能进行增删改
2. 每次读取都需要重新解析。