# 12-XML、Jsoup

一、XML
Extensible Markup Language 可扩展标记语言
可扩展：标签都是自定义的。 如<user> <student>.....

1.2、功能
存储数据
1、配置文件
2、在网络中传输。如发布的天气信息。

1.3、xml与html的区别
1、xml标签都是自定义的，html标签是预定义。
2、xml的语法严格，html语法松散
3、xml是存储数据的，html是展示数据
XML 是独立于软件和硬件的信息传输工具。

1.4、语法
1.4.1、基本语法：
1、xml文档的后缀名 .xml
2、xml第一行必须定义为文档声明
3、xml文档中有且仅有一个根标签
4、属性值必须使用引号(单双都可)引起来
5、标签必须正确关闭
6、xml标签名称区分大小写。大小写敏感。

<?xml version='1.0' ?> <users> <user id='1'> <name>zhangsan</name> <age>23</age> <gender>male</gender> <br/> </user> <user id='2'> <name>lisi</name> <age>24</age> <gender>female</gender> </user> </users>
代码示例

1.4.2、组成部分
1、文档声明
1. 格式：<?xml 属性列表 ?>
2. 属性列表：
1） version：版本号，必须的属性。1.1不向下兼容。
2）encoding：编码方式。告知解析引擎当前文档使用的字符集，默认值：ISO-8859-1
3）standalone：是否独立
4）取值：* yes：不依赖其他文件 * no：依赖其他文件

2、指令(了解)：结合css的 ---> 一般不展示数据
<?xml-stylesheet type="text/css" href="a.css" ?>
3、标签：标签名称自定义的
规则：
名称可以包含字母、数字以及其他的字符
名称不能以数字或者标点符号开始
名称不能以字母 xml（或者 XML、Xml 等等）开始
名称不能包含空格
4、属性：
id属性值唯一 ---> 并不是真正的id约束。需要加入约束文件。起的作用和HTML中的ID属性是一样的。
ID 仅仅是一个标识符，用于标识不同的便签。它并不是便签数据的组成部分。
XML 元素可以在开始标签中包含属性，类似 HTML。
属性 (Attribute) 提供关于元素的额外（附加）信息。
附注：

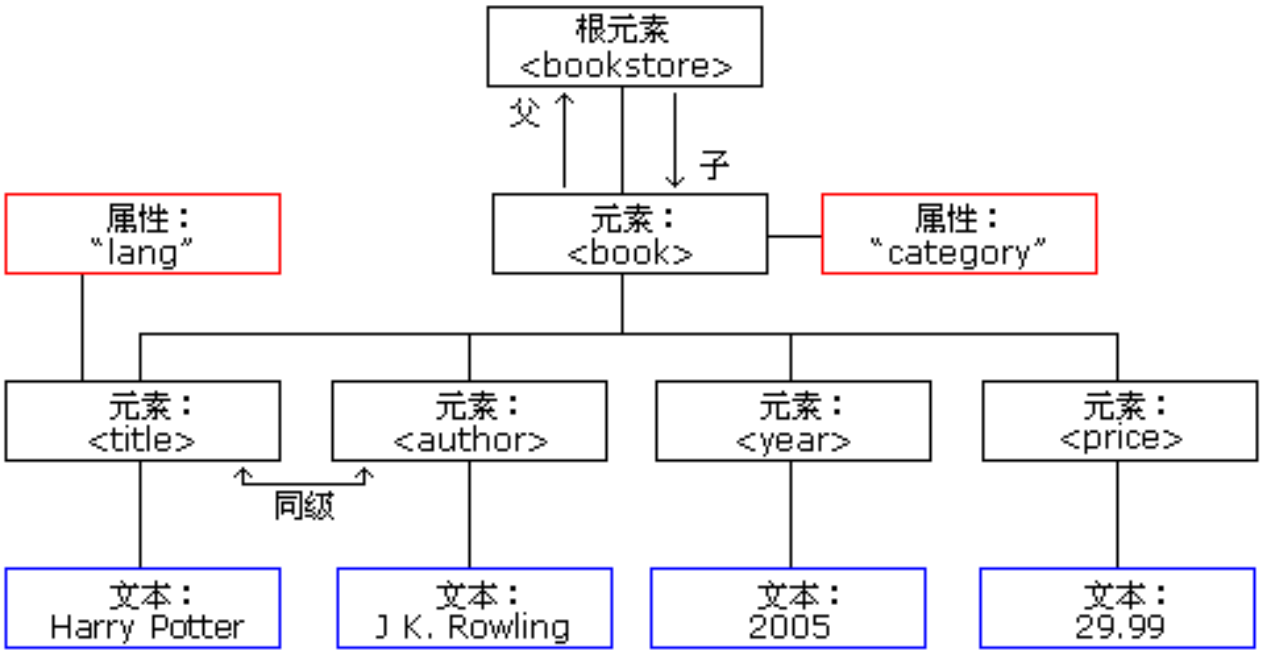
<person sex="female"> <firstname>Anna</firstname> <lastname>Smith</lastname> </person> <person> <sex>female</sex> <firstname>Anna</firstname> <lastname>Smith</lastname> </person>
View Code
XML元素VS属性
没有什么规矩可以告诉我们什么时候该使用属性，而什么时候该使用子元素。我的经验是在 HTML 中，属性用起来很便利，但是在 XML 中，您应该尽量避免使用属性。如果信息感觉起来很像数据，那么请使用子元素吧。
比较好的XML写法：

<note date="08/08/2008"> <to>George</to> <from>John</from> <heading>Reminder</heading> <body>Don't forget the meeting!</body> </note> <note> <date>08/08/2008</date> <to>George</to> <from>John</from> <heading>Reminder</heading> <body>Don't forget the meeting!</body> </note> <note> <date> <day>08</day> <month>08</month> <year>2008</year> </date> <to>George</to> <from>John</from> <heading>Reminder</heading> <body>Don't forget the meeting!</body> </note>
View Code
为什么避免使用XML属性：
1、属性无法包含多重的值（元素可以）
2、属性无法描述树结构（元素可以）
3、属性不易扩展（为未来的变化）
4、属性难以阅读和维护
5、文本：
CDATA区：在该区域中的数据会被原样展示 ==》相当于HTML中的<pre>
格式： <![CDATA[ 数据 ]]>

<code> <![CDATA[if(a<b&&a>c){}]]> </code> <code> if(a<b && a>c){} </code>
不写cdata区，需要使用转义字符。不便阅读。

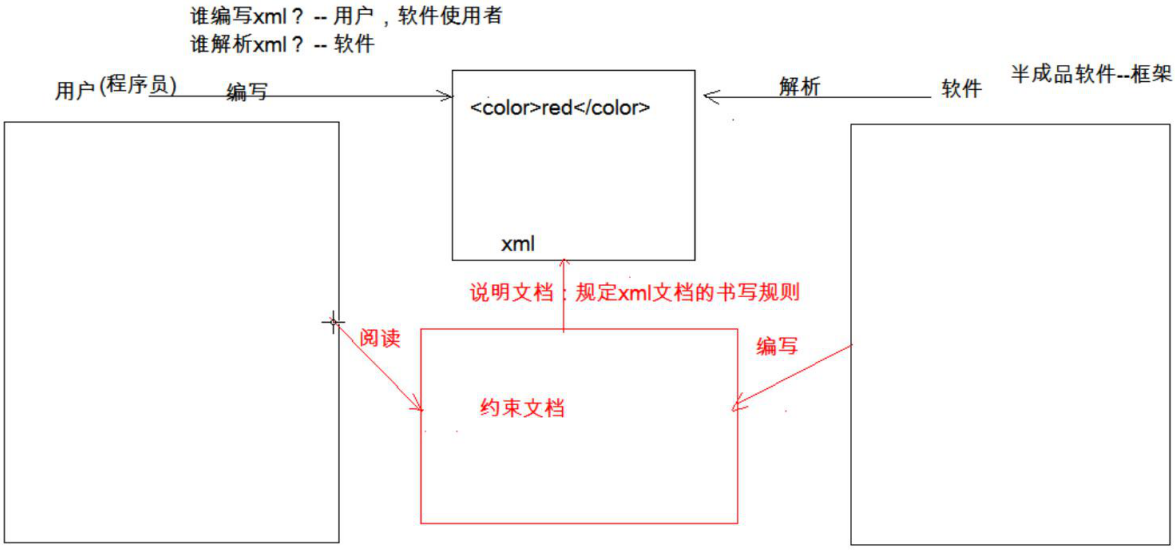
1.5、XML树结构

<bookstore> <book category="COOKING"> <title lang="en">Everyday Italian</title> <author>Giada De Laurentiis</author> <year>2005</year> <price>30.00</price> </book> <book category="CHILDREN"> <title lang="en">Harry Potter</title> <author>J K. Rowling</author> <year>2005</year> <price>29.99</price> </book> <book category="WEB"> <title lang="en">Learning XML</title> <author>Erik T. Ray</author> <year>2003</year> <price>39.95</price> </book> </bookstore>
View Code




XML元素
XML 元素指的是从（且包括）开始标签直到（且包括）结束标签的部分。
元素可包含其他元素、文本或者两者的混合物。元素也可以拥有属性。

二、约束
概念：规定xml文档的书写规则
附注：
xml标签可以随便定义。约束定义规定可以写那些标签。
避免产生歧义。信息描述不对等。书写规则。限定标签的书写。在什么地方该写什么标签。
按照要求来。
作为框架的使用者(程序员)：
1. 能够在xml中引入约束文档
2. 能够简单的读懂约束文档
分类：
1、DTD:一种简单的约束技术
2、Schema:一种复杂的约束技术
原理：



2.1、DTD
引入dtd文档到xml文档中。
==》存在缺陷：不能规定标签中的内容 age--1000 sex--hehe。而scheme可以进行内容上的限定。
使用一系列合法的元素来定义文档的结构。
1）内部dtd：将约束规则定义在xml文档中
2）外部dtd：将约束的规则定义在外部的dtd文件中
本地：<!DOCTYPE 根标签名 SYSTEM "dtd文件的位置">
网络：<!DOCTYPE 根标签名 PUBLIC "dtd文件名字(随意)" "dtd文件的位置URL"> ==》在某个服务器上存在该dtd文件的话。

<!--内部--> <!--<!DOCTYPE students <!ELEMENT students (student+) > <!ELEMENT student (name,age,sex)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ELEMENT sex (#PCDATA)> <!ATTLIST student number ID #REQUIRED> ]>-->
<!--外部--> <!ELEMENT students (student+) > //出现次数 <!ELEMENT student (name,age,sex)> //按照顺序出现一次 <!ELEMENT name (#PCDATA)> //字符串 <!ELEMENT age (#PCDATA)> <!ELEMENT sex (#PCDATA)> <!ATTLIST student number ID #REQUIRED> //<!--number属性名称 ID表示number值唯一 必须-->
使用：

<?xml version="1.0" encoding="UTF-8" ?> <!DOCTYPE students SYSTEM "student.dtd"> <!--<!DOCTYPE students [ <!ELEMENT students (student+) > <!ELEMENT student (name,age,sex)> <!ELEMENT name (#PCDATA)> <!ELEMENT age (#PCDATA)> <!ELEMENT sex (#PCDATA)> <!ATTLIST student number ID #REQUIRED> ]>--> <students> <student number="s001"> <name>zhangsan</name> <age>abc</age> <sex>hehe</sex> </student> <student number="s002"> <name>lisi</name> <age>24</age> <sex>female</sex> </student> </students>
View Code

2.2、Schema
可以进行更加细粒度的约束。约束的增强。
(在根标签)引入：
1.填写xml文档的根元素
2.引入xsi前缀. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3.引入xsd文件命名空间. xsi:schemaLocation="http://www.itcast.cn/xml student.xsd"
4.为每一个xsd约束声明一个前缀,作为标识 xmlns="http://www.itcast.cn/xml"

<students <!-- students 根标签 --> <!-- 固定格式 必须遵守的约束。约束约束文档的约束。 --><!-- 给别名再起一个别命名 如：xmlns:a="http://www.itcast.cn/xml" 此处使用默认 --> xmlns="http://www.itcast.cn/xml" <!-- xsi(根据xsi):schemaLocation(引入文件)= "http://www.itcast.cn/xml(需要使用的命名空间。该参数一个别名，唯一，可以写student。引入多个可能重复。域名或者倒置) ../dtd/student.xsd(本地。约束文件的位置。(也可是网络服务器上的文件))" --> xsi:schemaLocation(引入文件)="http://www.itcast.cn/xml student.xsd" > </students>

```
<?xml version="1.0" encoding="UTF-8" ?>
<a:students
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:a="http://www.itcast.cn/xml"
        xsi:schemaLocation="http://www.itcast.cn/xml  student.xsd">
    <!--<student number="heima_0001">
        <name>tom</name>
        <age>18</age>
        <sex>male</sex>
    </student>-->

    <a:student number="heima_0015">
        <a:name>payn</a:name>
        <a:age>18</a:age>
        <a:sex>male</a:sex>
    </a:student>
</a:students>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:mvc="http://www.springframework.org/schema/mvc"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <context:annotation-config />

    <context:component-scan base-package="cn.cisol.mvcdemo">
        <context:include-filter type="annotation"
            expression="org.springframework.stereotype.Controller" />
    </context:component-scan>
```

SpringMVC的配置文件：

<?xml version="1.0" encoding="UTF-8"?> <beans xmlns="http://www.springframework.org/schema/beans" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:context="http://www.springframework.org/schema/context" xmlns:mvc="http://www.springframework.org/schema/mvc" xsi:schemaLocation=" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc

http://www.springframework.org/schema/mvc/spring-mvc.xsd"> <context:annotation-config /> <context:component-scan base-package="cn.cisol.mvcdemo"> <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller" /> </context:component-scan> <mvc:annotation-driven /> <mvc:resources mapping="/resources/**" location="/resources/" /> <bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver"> <property name="order" value="1" /> <property name="mediaTypes"> <map> <entry key="json" value="application/json" /> <entry key="xml" value="application/xml" /> <entry key="htm" value="text/html" /> </map> </property> <property name="defaultViews"> <list> <bean class="org.springframework.web.servlet.view.json.MappingJackson2JsonView"> </bean> </list> </property> <property name="ignoreAcceptHeader" value="true" /> </bean> <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver"> <property name="viewClass" value="org.springframework.web.servlet.view.JstlView" /> <property name="prefix" value="/WEB-INF/jsps/" /> <property name="suffix" value=".jsp" /> </bean> <bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver"> <property name="maxUploadSize" value="209715200" /> <property name="defaultEncoding" value="UTF-8" /> <property name="resolveLazily" value="true" /> </bean> </beans>

View Code

URI对。成对出现。

xml格式 学习成本较低。dtd语法不一样

scheme可以设置取值的范围

namespace 引入的约束文件存在同名的标签。这是后将约束文件整体变成另一个命名空间。

解析框架完成。

## 三、解析

解析：操作xml文档，将文档中的数据读取到内存中

### 3.1、操作xml文档

1）解析(读取)：将文档中的数据读取到内存中

2）写入：将内存中的数据保存到xml文档中。持久化的存储

### 3.2、解析xml的方式：DOM思想 SAX思想

1）DOM：将标记语言文档一次性加载进内存，在内存中形成一颗dom树

优点：操作方便，可以对文档进行CRUD的所有操作

缺点：占内存

2）SAX：逐行读取，基于事件驱动的。

优点：不占内存。

缺点：只能读取，不能增删改

附注：

SAX解析思想：读一行释放一行。

基于事件的驱动。是开始的标签就会触发一些事件。知道了是什么标签。

服务器内存大一般使用DOM。移动端使用SAX占用内存小。

### 3.3、xml常见的解析器：

1、JAXP：sun公司提供的解析器，支持dom和sax两种思想。

2、DOM4J：一款非常优秀的解析器。

3、Jsoup：jsoup 是一款Java 的HTML解析器，可直接解析某个URL地址、HTML文本内容。它提供了一套非常省力的API，可通过DOM，CSS以及类似于jQuery的操作方法来取出和操作数据。

没有依赖的包。

4、PULL：Android操作系统内置的解析器，sax方式的。

解析进内存。得到dom树Document对象。

继承关系

Elemet-->Document

字符串通过转义字符进行嵌套。

## 四、Jsoup

### 4.1、Jsoup

jsoup 是一款Java 的HTML解析器，可直接解析某个URL地址、HTML文本内容。它提供了一套非常省力的API，可通过DOM，CSS以及类似于jQuery的操作方法来取出和操作数据。

4.2、使用步骤：
1、导入jar包
2、获取Document对象
3、获取对应的标签Element对象
4、获取数据

配置文件.xml放在src下。或者后期放在同一的文件夹下。
实例代码：

```
public class JsoupDemo01 { public static void main(String[] args) throws IOException { //1获取student.xml的path String path = JsoupDemo01.class.getClassLoader().getResource("com/itheima/xml/schema/student.xml").getPath(); //2解析xml文档，加载文档进内存，获取dom树--->Document Document document = Jsoup.parse(new File(path), "utf-8"); //3.获取元素对象 Elements Elements elements = document.getElementsByTag("a:student"); //4获取第一个name的Element对象 Element element = elements.get(0); System.out.println(elements.size()); //1 //5获取数据 System.out.println(element.text()); //payn 18 female } }
```
View Code

五、对象的使用
5.1、Jsoup：工具类，可以解析html和xml文档，返回Document。

parse：解析html或xml文档，返回Document 1）parse(File in, String charsetName)：解析xml或html文件的。 2）parse(String html)：解析xml或html字符串 3）parse(URL url, int timeoutMillis)：通过网络路径获取指定的html或xml的文档对象
5.2、Document：文档对象。代表内存中的dom树

获取Element对象 1）getElementById?(String id)：根据id属性值获取唯一的element对象 2）getElementsByTag?(String tagName)：根据标签名称获取元素对象集合 3）getElementsByAttribute?(String key)：根据属性名称获取元素对象集合 4）getElementsByAttributeValue?(String key, String value)：根据对应的属性名和属性值获取元素对象集合
5.3、Elements：
元素Element对象的集合。可以当做 ArrayList<Element>来使用

5.4、Element：元素对象

<1>获取子元素对象 1）getElementById?(String id)：根据id属性值获取唯一的element对象 2）getElementsByTag?(String tagName)：根据标签名称获取元素对象集合 3）getElementsByAttribute?(String key)：根据属性名称获取元素对象集合 4）getElementsByAttributeValue?(String key, String value)：根据对应的属性名和属性值获取元素对象集合 <2>获取属性值 String attr(String key)：根据属性名称获取属性值 <3>获取文本内容 1）String text():获取文本内容 2）String html():获取标签体的所有内容(包括子标签的字符串内容)

5.5、Node：节点对象
是Document和Element的父类。

六、快速查询方式
6.1、selector：选择器
方法：Elements select (String cssQuery) 参数:css选择器
语法：参考Selector类中定义的语法

6.2、XPath：XPath即为XML路径语言，是一种用来确定XML（标准通用标记语言的子集）文档中某部分位置的语言。
1、使用Jsoup的Xpath需要额外导入jar包。
2、查询w3cshool参考手册，使用xpath的语法完成查询

附注：
xpath-->路径语言
Jsoup对于xpath的支持度不高。只支持1.0的简单语法。JSoup不建议使用xpath。
dom4j可以使用。


## 公告