# XML

## 简介：

**XML:** **extensible Markup Language** 可扩展标记语言 version="1.0"

**可扩展**：所有的标签都是**自定义**的。

**功能：用于**数据存储、配置文件、数据传输

**html与xml区别**：

* html语法松散，xml语法严格
* html做页面展示，xml做数据存储
* html所有标签都是预定义的，xml所有标签都是**自定义的**

## xml语法

### 文档声明

|  |
| --- |
| 必须写在xml文档的第一行写法：<?xml version="1.0" ?>  属性：  version：版本号 固定值 1.0  encoding:指定文档的码表。一般写utf-8,也可以 iso-8859-1  standalone：指定文档是否独立 yes 或 no 【很少用】  如果是yes，则表示这个XML文档时独立的，不能引用外部的DTD规范文件；如果是no，则该XML文档不是独立的，表示可以用外部的DTD规范文档。 |

### 元素：xml文档中的标签

|  |
| --- |
| 文档中必须有且只能有一个根元素  元素需要正确闭合。<body></body> <br/>  元素需要正确嵌套  元素名称要遵守：  **元素名称区分大小写**  **数字不能开头** |

### 文本

|  |
| --- |
| 转义字符：&gt;  CDATA: 里边的数据会原样显示，不会产生歧义  **<![CDATA[ 数据内容 ]]>** |

### 属性

属性值必须用引号引起来。单双引号都行

### 注释

使用<!-- -->

### 案例：

|  |
| --- |
| <?xml version=*"1.0"* encoding=*"UTF-8"* standalone=*"yes"*?>  <stulist>  <student id=*"3"*>  <name>范冰冰</name>  <password>123</password>  <gender>女</gender>  <birthday>1983-10-11</birthday>  </student>  <student id=*"2"*>  <name>李冰冰</name>  <password>123</password>  <gender>女</gender>  <birthday>1984-10-11</birthday>  </student>  <student id=*"3"*>  <!-- <![CDATA[]]> 表示转义 -->  <name><![CDATA[&glt;>林心如<]]></name>  <password>123</password>  <gender>女</gender>  <birthday>1982-10-11</birthday>  </student>  </stulist> |

## XML约束

### 约束的分类

**dtd约束**或者**schema约束**

### dtd约束使用

第一步：创建一个student.dtd约束文件

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!ELEMENT students (student\*) >  <!ELEMENT student (name,password,gender,birthday)>  <!ELEMENT name (#PCDATA)>  <!ELEMENT password (#PCDATA)>  <!ELEMENT gender (#PCDATA)>  <!ELEMENT birthday (#PCDATA)>  <!ATTLIST student number ID #REQUIRED> |

* ATT:Attribute 属性的意思

第二步：在xml添加约束

|  |
| --- |
|  |
| **xml中引用dtd的方式**  1.本地dtd文件：<!DOCTYPE students SYSTEM "student.dtd">  2.网络dtd文件：<!DOCTYPE students PUBLIC "名称空间" "student.dtd"> |

### schema约束的使用

第一步：写个xsd

|  |
| --- |
| <?xml version=*"1.0"*?>  <xsd:schema xmlns=*"http://www.gyf.cn/xml"*  xmlns:xsd=*"http://www.w3.org/2001/XMLSchema"*  targetNamespace=*"http://www.gyf.cn/xml"* elementFormDefault=*"qualified"*>  <xsd:element name=*"students"* type=*"studentsType"*/>  <xsd:complexType name=*"studentsType"*>  <xsd:sequence>  <xsd:element name=*"student"* type=*"studentType"* minOccurs=*"0"* maxOccurs=*"unbounded"*/>  </xsd:sequence>  </xsd:complexType>  <xsd:complexType name=*"studentType"*>  <xsd:sequence>  <xsd:element name=*"name"* type=*"xsd:string"*/>  <xsd:element name=*"password"* type=*"xsd:string"* />  <xsd:element name=*"gender"* type=*"genderType"* />  <xsd:element name=*"birthday"* type=*"xsd:string"* />  </xsd:sequence>  <xsd:attribute name=*"number"* type=*"numberType"* use=*"required"*/>  </xsd:complexType>  <xsd:simpleType name=*"genderType"*>  <xsd:restriction base=*"xsd:string"*>  <xsd:enumeration value=*"male"*/>  <xsd:enumeration value=*"female"*/>  </xsd:restriction>  </xsd:simpleType>    <xsd:simpleType name=*"numberType"*>  <xsd:restriction base=*"xsd:string"*>  <xsd:pattern value=*"itgyf\_\d{4}"*/>  </xsd:restriction>  </xsd:simpleType>  </xsd:schema> |

第二步：写个xml引用xsd

|  |
| --- |
| 导入xsd约束文档：  1.编写根标签  2.引入实例名称空间 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  3.引入名称空间 xsi:schemaLocation="http://www.gyf.cn/xml student.xsd"  4.引入默认的名称空间 |
| <?xml version=*"1.0"* encoding=*"UTF-8"*?>  <students  xmlns=*"http://www.gyf.cn/xml"*  xsi:schemaLocation=*"http://www.gyf.cn/xml student.xsd"*  xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*>  <student number=*"itgyf\_1001"*>  <name>gyf</name>  <password>123</password>  <gender>male</gender>  <!-- 这是有顺序的 -->  <birthday>2017-11-12</birthday>  </student>  </students> |

# XML解析

## 简介

xml解析就是读写xml的意思

## XML解析思想【面试题】：

### DOM解析：

将文档加载进内存，形成一颗dom树(document对象)，将文档的各个组成部分封装为一些对象。

优点：因为，在内存中会形成dom树，可以对dom树进行增删改查。

缺点：dom树非常占内存，解析速度慢。

### SAX解析

逐行读取，基于事件驱动

优点：不占内存，速度快

缺点：**只能读取，不能回写(改数据)**

## XML常用的解析器

### **JAXP**：

* + sun公司提供的解析。支持dom和sax。JAXP（Java API for XMLProcessing，意为XML处理的Java API）,它提供解析和验证XML文档的能力。

### **JDOM**：

* + JDOM是一个开源项目，它基于树型结构，利用纯JAVA的技术对XML文档实现解析、生成、序列化以及多种操作。

### **DOM4J**：

dom for java

* + dom4j是一个Java的XML API，**是jdom的升级品**，用来读写XML文件的。dom4j是一个十分优秀的JavaXML API，具有性能优异、功能强大和极其易使用的特点，它的性能超过sun公司官方的dom技术

# DOM4J解析

## DOM4J的SAX使用

根据文档来写

**第一步：**导入包

|  |
| --- |
|  |

**第二步：**遍历节点

@Test 单元测试

|  |
| --- |
|  |

## XPATH：

专门用于查询,它定义了一种规则。

### 使用的方法：

selectSingleNode():

selectNodes():

### 使用步骤：

1、**注意：要导包 jaxen...jar**

2、创建解析器SAXReader reader = new SAXReader();

3、解析xml 获得document对象 Document document = reader.read(url);

### XPath规则：

|  |
| --- |
| **// nodename 选取此节点。**  **/ 从根节点选取。**  **.. 选取当前节点的父节点。**  **@ 选取属性。**  [@属性名] 属性过滤  [标签名] 子元素过滤 |
|  |

## 修改节点保存

|  |
| --- |
| **public** **static** **void** main(String[] args) **throws** Exception {  SAXReader reader = **new** SAXReader();    Document doc = reader.read(**new** FileInputStream("xml/books.xml"));  List list = doc.selectNodes("//书/@出版社");    **for**(Object o : list){  **if**(o **instanceof** Attribute){  Attribute att = (Attribute) o;  att.setValue("美国");  }  }    XMLWriter writer = **new** XMLWriter(**new** FileOutputStream("xml/books.xml"));  writer.write(doc);  } |