# Xml

## 需求

XML：X- M- L- 跟html书写习惯非常类似。

现在有一套页面,这套页面被其他人访问到,端口号可以进行修改(不修改源码的基础上).

## 分析

### 2.1技术分析

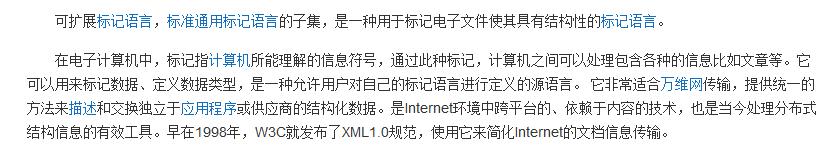
1. java基础的网络编程

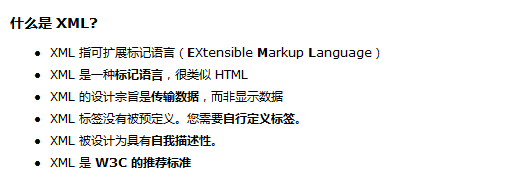
xml也是常见的数据交换格式:AJAX

1. xml 配置文件使用
2. XML用作小型数据库

## xml概述

### 3.1什么是xml?





HTML编写的时候，核心骨架：

<HTML>

<HEAD>

<BODY></BODY>

</HTML>

允许用户自定义标签!!!

### 3.2xml的作用

1. 传输和存取数据
2. 软件的配置文件

## xml的语法

### xml的基本语法

Xml必须有关闭标签 <START></START>

区分大小写 <a> <A>

属性需要有引号 <a id=”23”></a>

标签必须正确的嵌套

举例：不正确的嵌套格式

<parent>

<son>

</parent>

</son>

必须有根元素

类似Html文档里面<html>

### 2.xml的文档声明

文档声明: 通常出现在xml的第一行第一列的位置!!!

写法:

<?xml 属性名=”属性值” 属性名=”属性值” ...>

version: 版本号,必须的 默认值1.0

encoding: 编码字符集,是使用浏览器打开的时候采用的默认的字符集的编码.

standalone: 描述xml文档是否需要依赖其他的文件.yes/no

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

<根元素>

。。。

</根元素>

### 3.xml的注释

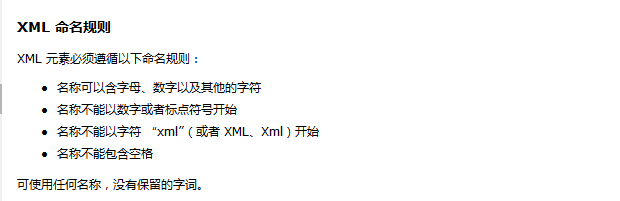
<!-- xml的注释 -->

### xml的元素

xml中的元素理解java类中的属性。元素命名规则也可以遵循java属性命名规则

元素命名：建议全大写。习惯camel命名。

元素(标签)的命名规范:



名称中不能包含冒号 :

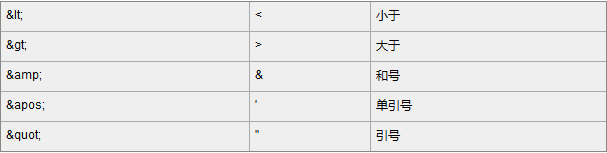
### 5.xml的属性

属性的名称规范与元素一致.

属性需要使用引号!!!

### 6.xml的特殊字符和CDATA区

xml的特殊字符



xml的CDATA区: CDATA -------------Character Data

语法:

<![CDATA[ 内容 ]]>

## xml的解析

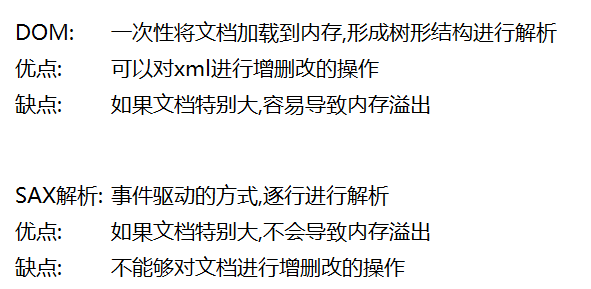
xml的解析: 从xml文档中获取想要的数据(通过代码完成的)

xml的解析方式:

DOM解析: document object model文档对象模型方式

SAX解析: simple api for xml.

DOM和SAX的区别:



针对这两种解析方式,不同的公司提供了不同的api实现.

第三方：非java提供。

JAXP: SUN公司提供的一套xml的解析的api

JDOM: 开源组织提供的一套xml的解析的api-jdom.

DOM4J: Apache开原组织提供的一套xml的解析的api-dom4j.

Dom for java🡺dom4j

Log4j🡺日志记录

Tomcat

DBUtils

数据库连接池

PULL: 主要应用在Android移动端解析xml

### 5.1Dom4J入门案例

步骤:

1. 导入jar包.资料下的dom4j-1.6.1.jar
2. 创建解析器
3. 解析文档获得代表文档的document对象
4. 获取根节点
5. 通过根节点向下查找其他的节点

代码:

@Test

/\*

\* 获取元素的内容:查询的操作

\*/

**public** **void** test1() **throws** Exception {

// 创建解析器

SAXReader reader = **new** SAXReader();

// 解析xml文档,得到document对象

Document document = reader.read("xml/demo1.xml");

// 根据document对象获取根节点

Element root = document.getRootElement();

System.***out***.println(root.getName());

// 查找根节点下的子节点 element() elements()

// Element p1 = root.element("person"); // 查找的是第一个person元素

Element p2 = (Element) root.elements("person").get(1);// 查找的是第二个person元素

Element name = p2.element("name");

Element age = p2.element("age");

Element sex = p2.element("sex");

System.***out***.println(name.getText());

System.***out***.println(age.getText());

System.***out***.println(sex.getText());

}

### 5.2XPath案例

要引入Dom4j支持xPath的jar包

jaxen-1.1-beta-6.jar

Dom4j的xPath支持的API:

List document.selectNodes(String xPath);

Node document.selectSingleNode(String xPath);

代码:

@Test

/\*

\* 使用xPath简化查询

\*/

**public** **void** test2() **throws** Exception {

// 创建解析器

SAXReader reader = **new** SAXReader();

// 解析xml文档,返回document对象

Document document = reader.read("xml/demo1.xml");

/\*

\* List<Node> list = document.selectNodes("//name"); for (Node node :

\* list) { Element element = (Element) node;

\* System.out.println(element.getText()); }

\*/

List<Node> list = document.selectNodes("//person['@id']");

**for** (Node node : list) {

Element element = (Element) node;

System.***out***.println(element.attributeValue("id"));

}

}

## 6.xml作为配置文件模拟tomcat案例

步骤:

1. 将se阶段socket编写的模拟tomcat的代码复制到工程
2. 新建一个xml文件作为配置文件,放在conf/server.xml
3. 使用dom4j解析xml

代码:

**public** **static** **void** main(String[] args) **throws** Exception {

// 解析xml文件

// 创建解析器

SAXReader reader = **new** SAXReader();

// 解析xml

Document document = reader.read("conf/server.xml");

Element ele = (Element) document.selectSingleNode("//Connector['@port']");

String port = ele.attributeValue("port");

// ServerSocket 对象可以监听端口

ServerSocket serversocket = **new** ServerSocket(Integer.*parseInt*(port));

**while** (**true**) {

Socket socket = serversocket.accept(); // 等待客户端的连接请求，一旦有请求过来，就结束阻塞，返回客户端对象

// System.out.println(socket.getInetAddress());

// 一旦有客户来访问， 就另开一个新线程去提供服务， main线程继续等待下一个客户的连接

**new** Thread(**new** MyService(socket)).start();

}

}

## xml的约束

### 7.1xml约束概述

什么是xml的约束: 就是用来约束xml文档中可以出现哪些标签,不能出现哪些标签,标签 是否有顺序,标签出现的次数.

xml约束的作用: 用来规范xml的写法

xml的约束的种类和区别:

种类: DTD约束 schema约束

区别:

1. DTD语法是自成一体的.schema语法就是xml的语法
2. Schema的语法就是xml的语法,所以更容易被解析器解析.
3. Schema约束支持名称空间.
4. Scheme约束有比DTD更加强大的语义和语法的约束

### 7.2DTD约束的语法(了解)

DTD的引入方式:

内部的DTD:

<!DOCTYPE 根标签 [

]>

外部的DTD:

本地的DTD:

<!DOCTYPE 根标签 SYSTEM "unknown.dtd">

网络的DTD:

<!DOCTYPE 根标签 PUBLIC "//UNKNOWN/" "unknown.dtd">

DTD的语法

元素:

<!ELEMENT 元素名称 元素类型>

元素类型:

EMPTY

ANY

子元素

是否有序: 使用逗号(,)或者使用竖线(|)

出现的次数: ?:零次或一次 +:一次或多次 \*:零次或多次

PCDATA

属性:

<!ATTLIST 元素名称 属性名称 属性类型 属性约束>

属性类型:

ID类型: 表示属性值需要是唯一的 id,name=”aa”

CDATA类型: 普通的字符串 name=”aa”

枚举: target=”\_blank,\_self”

属性约束:



### 7.3DTD约束示例代码

<!ELEMENT persons (person\*)>

<!ELEMENT person (name|age|sex)+>

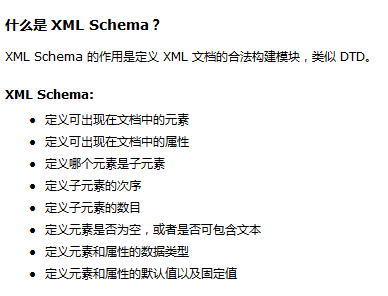
<!ELEMENT name (#PCDATA)>

<!ELEMENT age (#PCDATA)>

<!ELEMENT sex (#PCDATA)>

<!ATTLIST person id ID #REQUIRED>

### 7.4schema约束(了解)



<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<!--

名称空间:一个xml只能引入一个DTD的约束文档,使用schema去约束xml文档,一个xml可以引入多个scheme的约束!!!

如果有多个schema约束,里面定义了相同的属性名称,该怎么办?

名称空间类似于java中的package,通过名称空间来区分我们的标签和属性是来自于那个文档的!!!通常名称空间的名称是唯一不重复的即可

一般情况下,使用一个url地址来表示一个名称空间

xmlns: xml name space 代表当前的文档应用的名称空间

targetNamespace: 目标的名称空间

elementFormDefault:

-->

<schema

xmlns=*"http://www.w3.org/2001/XMLSchema"*

targetNamespace=*"http://www.whhp.com/aaa"*

elementFormDefault=*"qualified"*>

<!-- 复杂标签 -->

<element name=*"persons"*>

<!-- 复杂类型 -->

<complexType>

<sequence maxOccurs=*"unbounded"* minOccurs=*"1"*>

<element name=*"person"*>

<!-- 复杂类型 -->

<complexType>

<sequence>

<!-- 简单标签 -->

<element name=*"name"* type=*"string"*></element>

<element name=*"age"* type=*"integer"*></element>

<element name=*"sex"* type=*"string"*></element>

</sequence>

<attribute name=*"id"* type=*"string"* use=*"required"*></attribute>

</complexType>

</element>

</sequence>

</complexType>

</element>

</schema>

### 7.5编写schema约束的流程

