1. XML
2. 概述

Extensible Markup Language，可扩展标记语言。所谓的可扩展性指的是XML的标签都是自定义的，不像HTML中标签都是预定义的。

1. 功能

XML是用于存储数据的，一般用于配置文件和网络中传输。

1. XML与HTML的区别
2. XML标签是自定义的，HTML标签是预定义的
3. XML的语法严格，HTML的语法松散
4. XML用于存储数据，HTML用于展示页面
5. XML的语法
6. 基本语法
7. XML文档中有且仅有一个根标签
8. XML文档第一行为XML文档声明
9. XML文档的后缀名为.xml
10. XML文档的标签属性名需用引号引起来
11. XML标签区分大小写
12. XML文档的组成
13. 文档声明
14. 格式

<?xml 属性列表 ？>

1. 属性列表

version:版本号，必写

encoding:编码方式，告知解析引擎当前文档使用的字符集

standalone:：是否独立，独立即不依赖其他文件

1. 标签

自定义的标签命名规则：

1. 名称不能以数字或者标点符号开始
2. 名称可以包含字母，数字以及其他字符
3. 名称不能以xml（XML）开始
4. 指令

结合CSS控制样式

1. 文本

CDATA区：能够将该区域中的数据原样展示出来，如：

<![CDATA[数据]]>

1. XML文档的约束
2. 概述

XML文档受到约束文档的限制，通过约束文档来规范XML文档的书写规则。将约束文档引入XML文档中。

1. 约束的分类
2. DTD

dtd文档的简单语法：

通过Element来定义标签

如：定义了一个students标签，该标签中有多个student标签。

<!ELEMENT students(student\*)>

如：定义一个student标签，该标签还有 name，age，sex三个标签，每个标签只能按照顺序出现一次。

<!ELEMENT student(name, age, sex)>

如：定义一个name标签，该标签中存储的是字符串

<!ELEMENT name(#PCDATA)>

dtd文档的导入：

将dtd文档引入到XML文档中，分为两种方式：

一种是写在XML文档内部的dtd

一种是写在XML文档外部的dtd文档。

外部的dtd文档又分为本地文档与网络中的文档：

本地dtd：<!DOCTYPE 根标签名SYSTEM “dtd文件的位置”>

根标签名为该dtd文档约束的XML文档中的跟标签名，即最外层的标签名，SYSTEM代表的是本地的dtd文档

网络dtd：<!DOCTYPE 根标签名 PUBLIC “dtd文件名字” “dtd文件的位置URL”>

同理，PUBLIC 代表着网络中的dtd文件

1. Schema(后缀为xsd)

xsd文档的简单语法；

xsd文档能够更加详细地限定XML的标签中的内容，当然也是更加的复杂。本质上xsd文档还是xml文档。

如：定义一个标签元素 students，类型是自定义类型studentsType

<xsd:element name=”students” type=”studentsType” />

如：为students标签定义自定义类型studentsType，该类型中限定按照顺序存储student标签，student标签又限定为自定义studentType类型，最小值为0，最大值未限定。

<xsd::complexType name=”studentsType”>

<xsd:sequence>

<xsd:element name=”student” type=”studentType” minOccurs=”0” maxOccurs=”unbounded” />

</xsd:sequence>

</xsd:complexType>

接下来，为student定义studentType类型(复杂类型)，按照顺序存储name标签（类型为string），age标签(类型为ageType)，sex标签(类型为sexType)，最后限定一个必选的属性number，类型为自定义numberType。

<xsd:complexType name=”studentType”>

<xsd:sequence>

<xsd:element name=”name” type=”xsd:string” />

<xsd:element name=”age” type=”ageType” />

<xsd:element name=”sex” type=”sexType” />

</xsd:sequence>

</xsd:attribute name=”number” type=”numberType” use=”required” />

</xsd:complexType>

以ageType举例，ageType为简单类型，该类型限定基本数据类型为integer，最小值为0，最大值为256.

<xsd:simpleType name=”sexType”>

<xsd:restriction base=”xsd:integer”>

<xsd:minInclusive value=”0” />

<xsd:maxInclusive value=”256”/>

</xsd:restriction>

</xsd:simpleType>

xsd文档的导入：

1. 填写XML文档的根标签students
2. 引入xsi前缀，xmlns:xsi=””
3. 引入xsd文件命名空间 xsi:schemaLocation=””
4. 为每一个xsd约束空间声明一个前缀，作为标识，因为当多个xsd文件中有相同的标签名需要区别，使用前缀会更加的方便。如：将该xsd声明为a

xmlns:a=””

如：

<students xmlns:xsi=<http://www.w3.org/2001/XMLSchema-instance>

xsi:schemaLocation=”./xml student.xsd”

xmlns:a=”./xml”

>

1. XML的解析
2. 概述

操作XML文档，将文档中的数据读取到内存中，或者将内存中的数据保存到XML文档中，持久化地存储。

1. 解析XML的方式
2. DOM

将标记语言文档一次性地加载到内存中，在内存中形成一个DOM树。

1. 优点

操作非常方便，可以对文档进行增删改查的操作

1. 缺点

缺点是DOM树是一开始就会生成的，且非常占内存，因此一般用于服务器端。

1. SAX

不是一次性地加载到内存中，而是逐行读取。

1. 优点

不会占用太大的内存，因此经常用于移动端等内存不大的环境中。

1. 缺点

只能读取，不能增删改操作。

1. XML的常见解析器
2. JAXP

sun公司提供的解析器，支持DOM和SAX两种思想

1. DOM4J
2. Jsoup

Java的一款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。

1. Jsoup解析器
2. Jsoup的使用步骤
3. 导入Jsoup的jar包
4. 获取Document对象
5. 获取对应的标签Element对象
6. 获取数据
7. 对象的使用
8. Jsoup

工具类，可以解析HTML或XML文档，返回Document对象

Jsoup.parse(File in, String charsetName):解析xml或HTML文件

Jsoup.parse(String html):解析html或xml字符串

Jsoup.parse(URL url, int timeoutMillis):通过网络路径获取指定的HTML或xml的文档对象

1. Document

文档对象，代表内存中的DOM树，获取Element对象的方法如下所示：

getElementById(String id):根据id属性值获取唯一的element对象

getElementsByTag(String tagName):根据标签名称获取元素对象集合

getElementsByAttribute(String key):根据属性名称获取元素对象集合

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

1. Elements

Element对象的集合，可当做ArrayList<Element>来使用

1. Element

元素对象，在该元素对象中使用如下方法获取子元素对象：

getElementById(String id):根据id属性值获取唯一的element对象

getElementsByTag(String tagName):根据标签名称获取元素对象集合

getElementsByAttribute(String key):根据属性名称获取元素对象集合

getElementsByAttributeValue(String key, String value)：根据对应的属性名和属性值获取

在该元素对象中使用如下方法获取属性值：

String attr(String key)：根据属性名称获取属性值

在该元素对象中使用如下方法获取文本内容：

String text():获取子元素对象中的所有纯文本内容

String html():获取子元素对象中的所有内容（包括子元素的标签和纯文本内容）

1. Node

节点对象，

1. Jsoup选择器查询
2. selector:选择器

通过选择器可以查找出文档中的标签信息，属性信息以及文本信息。

Elements select(String cssQuery)

如：

查找document中标签名为name的元素  
document.select(“name”);

查找document中id值为a的元素

document.select(“#a”);

查找doocument中的student标签，且该标签的number属性值为a的子标签

document.select(“student[number=\”a\”]”);

1. Xpath

Xpath为XML的路径语言，是一种用来确定XML文档中某部分位置的语言，使用Jsoup的Xpath需要额外导入Xpath的jar包。通过查询手册，使用Xpath的语法查询。

如：

通过标签名查询，标签名前是使用//  
List<JXNode> jxNodes = jxDocument.selN(“//student”);

查询student标签下的子标签name:，子标签中间用/隔开

List<JXNode> jxNodes = jxDocument.selN(“//student/name”);

查询student标签下id属性等于a的name标签，使用@表示id

LIst<JXNode> jxNode = jxDocument.selN(“//student/name[@id=’a’]”);