1. 表单的提交方式
2. 使用submit提交

<form>

……..

<input type =”submit”/>

</form>

1. 使用button表单提交

function forml(){

//获取form

var forml=document.getElementById("forml");

//设置action

forml.action="hello.html";

//提交form表单

form1.submit()；

}

1. 使用超链接提交

<a href=”hello.html?username=123456”>使用超链接提交</a>

1. button 的事件
2. onclick 鼠标点击事件
3. onchange:改变内容（一般和select一起使用）
4. onfocus：得到焦点（ie5）
5. onblur：失去焦点

<input type="text" id="id1" name="text1" value="please input" onfocus="focus1();" onblur="blur1();"/>

<script type="text/javascript"/>

function focus1(){

var input1 = document.getElementById(“id1”);

iinput1.value=””;点击进去为空值

}

function blur1(){

var input1 = document.getElementById(“id1”);

iinput1.value=” please input”;点击进去为空值

}

</script>

类似搜索中 里边有文字，点进去之后里边没有文字

# xml的简介

* 1. 可扩展标记型语言（extensible Markus language）

XML是一种可扩展的标记语言，它被设计用来传输和存储数据，是由万维网协会推出的一套数据。它可以用于定义Web网页上的文档元素，以及复杂数据的表述和传输。XML与HTML的主要差异在于：HTML用来显示数据，XML用来传输和存储数据；HTML用来显示信息，XML用来传输信息。XML的最大特点是它的自我描述和任意扩展，当用其描述数据时，用户可以根据需要，组织符合XML规范形式的任意内容，并且标签的名称也可以由用户指定。下面以例子来说明XML的

交换标准

标记型语言：html标记型语言

也是使用标签来操作

可扩展：html里面的标签是固定的，每个标签都有特定的含义

<h1><br/><hr/>

标签可以使自己定义，可以中文的标签<person><猫>

* 1. xml的用途：
     1. 传输数据

通过XML可以在不同的系统之间传输数据，在开发过程中难免会遇到多个系统之间相互通信，且各系统的存储数据又是多种多样的情况，对于开发者而言，这些工作量是巨大的，通过转换为XML格式来传输数据可以减少传输数据时的复杂性，并且还可以具备通用性。

2、存储数据

利用XML来存储数据是其最基本的用途，因为它可以作为数据文件，所以当需要持久化保存数据时，可以利用XML数据格式进行存储，例如，web.xml、struts.xml、spring.xml等。下面就是经常见到的web.xml文件内容：

xml是w3c组织发布的一个技术

版本 1.0 和1.1 （使用的一般为1.0版本 1.1向下不兼容）

3、xml的应用

不同的系统之间传输数据

qq之间传输数据

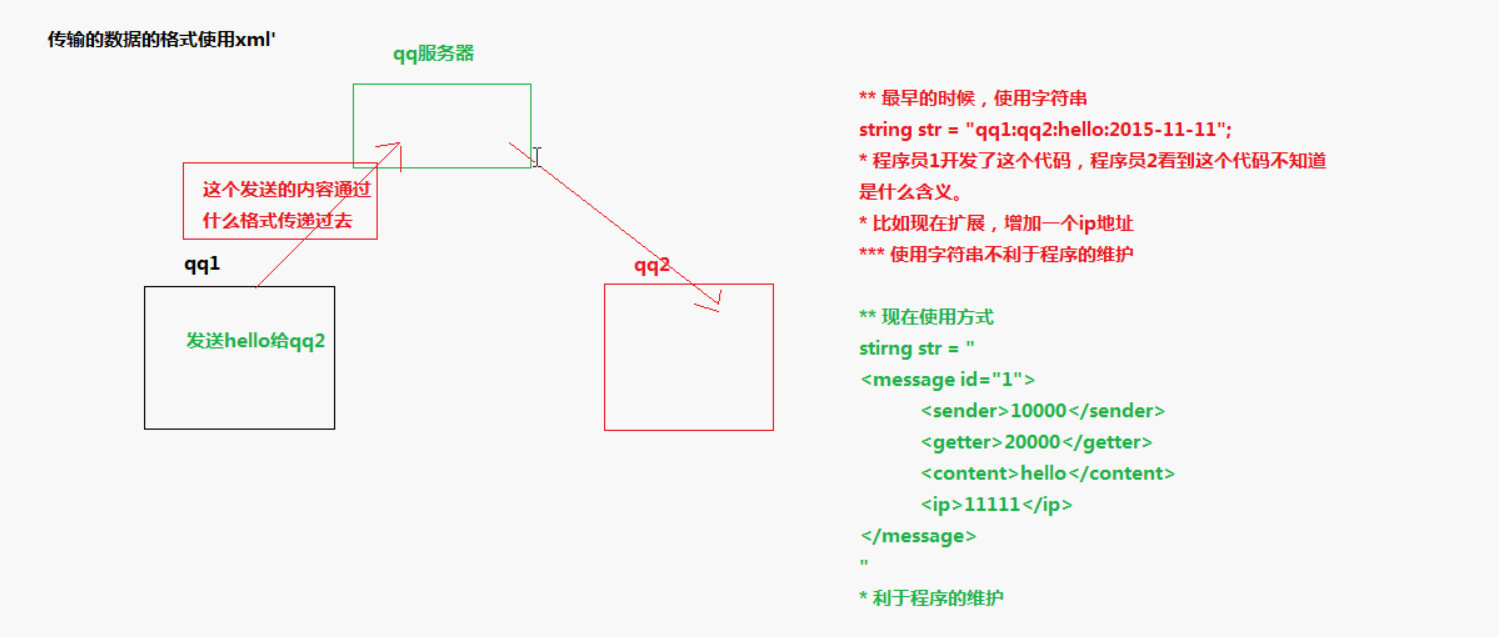
用来表示生活中有关系的的数据

例如：山西省-太原市-迎泽区

经常用在配置文件、

比如现在连接数据库：知道数据库的用户名和密码 数据库名称

如果修改数据的信息，不需要修改源代码，只需要修改配置文件

数据的传输格式

* 1. xml的技术架构
     1. xml的文档声明

创建一个文件 后缀.xml

如果写xml，第一步必须有一个文档说明（写 了文档声明之后，表示写xml里面的东西）

<?xml version=”1.0” encoding=”gbk”?>文档声明

注意点：位置：必须为第一行第一列

属性：version xml的标准版本 1.0 1.1 两个版本

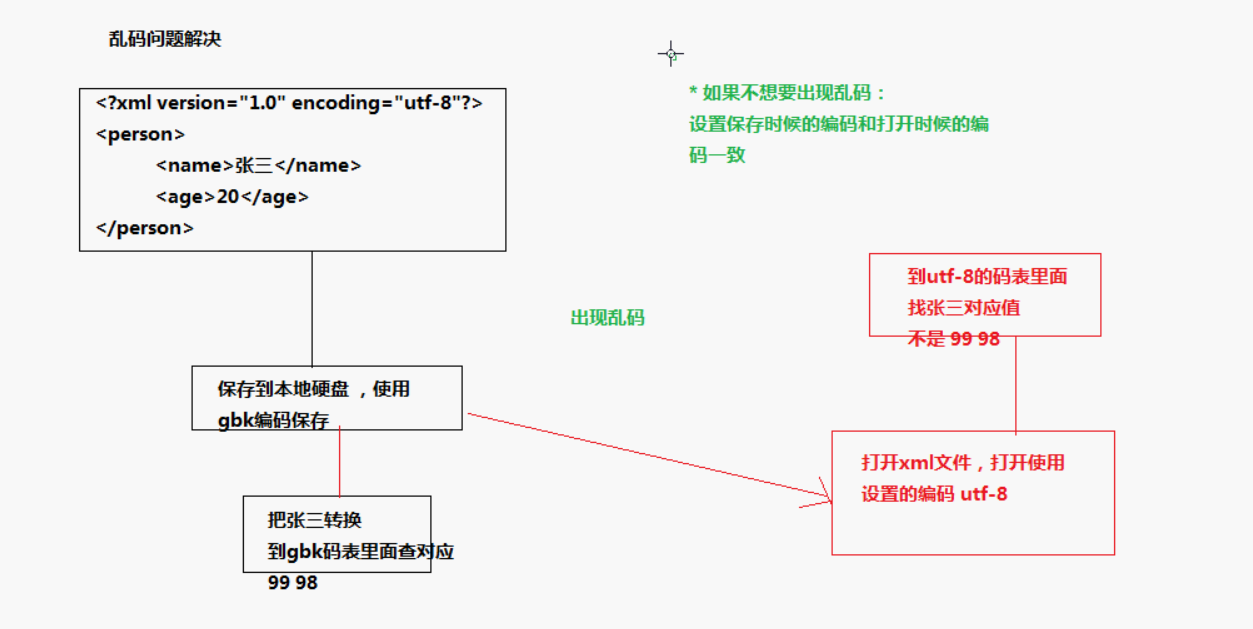
Encodeing：指明文档使用的字符编码格式，xml编码 gbk中文简体 UTF-8 iso8859-1（不包含中文

Standalone：是否需要依赖其他的文件 值 yes no

（每个属性之间必须有空格）

Xml的中文乱码

乱码分析



保存时候的编码和大家时候的编码要一致

* 1. xml的元素的定义（标签的定义）

标签定义<person></person>

* + 1. 标签成对形式出现。
    2. 标签没有内容，可以在标签内结束<a/>
    3. 标签可以嵌套，但是必须要合理嵌套

<dad><son></son></dad>即dad是son的父元素son是dad的子元素

<aa><bb></aa></bb>不合理嵌套

4、一个xml有且只有一盒根标签，其他的标签都是这个标签这么下面的标签

5、在xml中，它会将空格和换行内容进行解析

下面中的代码含义是不一样的

<aa>111111</aa>

<aa>

1111

</aa>

* 1. xml中标签的命名规则
     1. xml代码中是区分大小写的。<age><Age>是两个不同的标签。
     2. 标签名称不能以数字和下划线开头，只能 以字母和中文字开头
     3. 不能以xml 、XML、Xml 开头 涉及xml的都不可以 都是不正确的
     4. xml标签里面不能包含空格和冒号
     5. xm的标签可以是中文
  2. xml中属性的定义

\*htm1是标记型文档，可以有属性

\*xm1也是标记型文档，可以有属性

<person></person>

定义要求：1、一个标签上可以有多个属性。

<>111111</aa>

2、属性名称不能相同。

3属性名和属性之间使用“=”属性值说那个引号抱起来（可以使单引号，也可以是双引号）

4属性值必须使用引号，不过单引号和双引号均可使用，如果属性值本身包含双引号，那么有必要使用单引号包围它或者可以使用实体（&quot）引用它。

1. xml的名称规范和元素的规范是一样的

7、xml中的注释

写法 <!—注释内容-->

注：注释不可以有嵌套

1. xml中的空白被保留

空白被保留是指在XML文档中，空白部分并不会被解析器删除，而是被当作数据一样完整的保留。

1. xml中中的特殊字符

如果想要在xml中显示 a<b 不能正常显示，因为把<当做标签

如果就想要显示，需要对特殊字符<进行转义

@ ---@amp

< --&lt

>---&gt

“ ---&quot

‘ --- &apos

1. CDATE区:可以解决有多个字符都需要转义的需求，将这些放到CDATE区里

CDATA是用于需要原文保留的内容，尤其是在解析XML过程中产生歧义的部分，当某个节点的数据有大量需要转义的字符时，那么CDATA就可以发挥其作用。

写法<![CDATE [

内容

]]>

1. PI指令（处理指令）

可以再xml中设置样式

<?xml-stylesheet type="text/css" href="1.js"?>

Xml：

<? xml version="1.0" encoding="utf=8">

<?xml-stylesheet type="text/css" href="1.js"?>

<person>

<name>1111111111</name>

<age>222222222222</age>

</person>

Js中

name{

background:red;

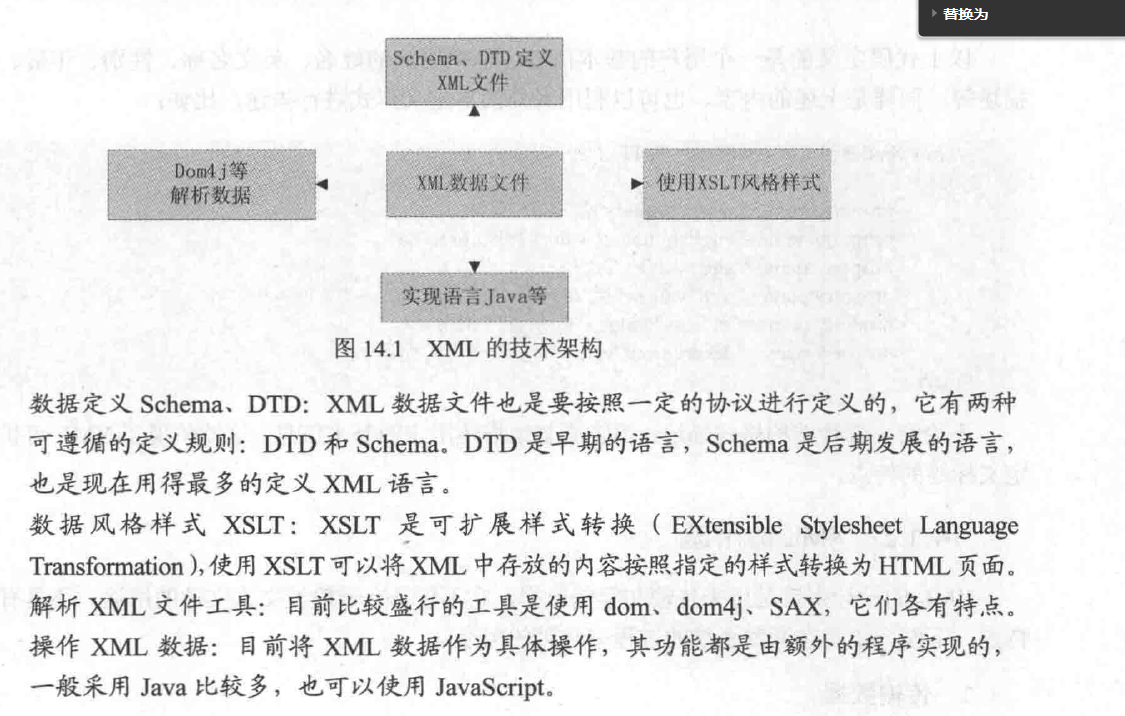
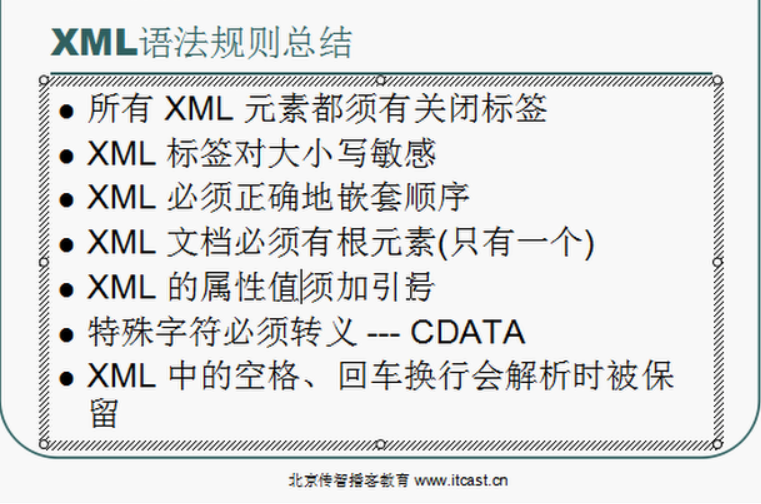
}

age{

background:red;

}

PI设置的样式只能对英文标签起作用，中文标签不起作用



11、xml的约束

为什么需要约束？

比如现在定义一个person的xml文件，只想要这个文件保存人的信息，比如 name age等，但是，如果在xml文件中写了一个标签<猫>，发现可以正常显示，应为符合语法规范。但是猫肯定不是人的信息，新买来的标签是自定义的，需要技术来规定xml只能出现的元素，这个时候需要约束

xml的约束技术：dtd的约束和schema约束 （常见的约束）

12、dtd的快速入门

创建一个文件 后缀名 .dtd

步骤 （1）看xml中有多少个元素，有几个元素，在dtd文件文件写几个<!ELEMENT>

(2)判断元素是简单元素还是复杂元素

复杂元素：有子元素、

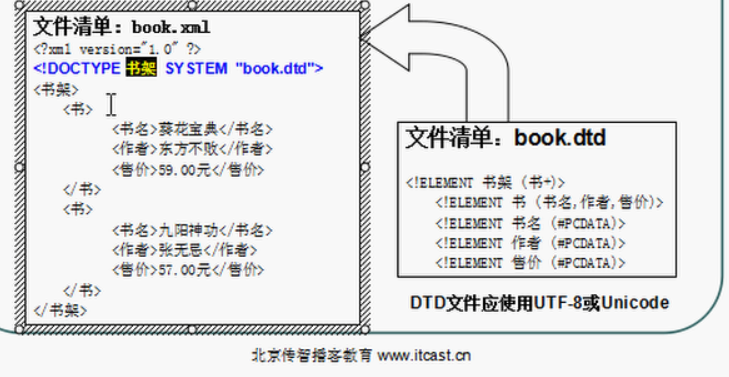
<!ELEMENT 元素名称 (子元素) >

简单元素：没有子元素

<!ELEMENT 元素名称 (#PCDATE) >

(3)在xml文件中引入dtd文件

<!DSOCTYPE 根元素名称 SYSTEM “dtd 文件的路径”>



打开xml文件使用浏览器打开的，浏览器只负责校验语法，而不负责校验约束

如果校验xml的约束，需要使用工具

\*\*打开myeclipse开发工具

\*\*\*创建一个项目dayo5

\*\*\*在day05的src目录下面创建一个xm1文件和一个dtd文件

\*\*\*当xml中引入dtd文件之后，比如只能出现name age，多写了一个a，会提示出错

13、dtd的引入方式（三种）

1、引入外部的dyd文件

<!DOCTYPE 根元素名称 SYSTEM “根元素的路径”>

2、使用内部的dtd代码（在xml中使用dtd）

-<!DOCTYPE 根元素名称{

<!ELEMENT person (name,age)>

<!ELEMENT name(#PCDATE)>

<!ELEMENT name(#PCDATE)>

}>

3、使用外部的dtd文件（网络上的dtd文件）

<!DOCTYPE 个人元素 PUBLIC “DTD名称” “DTD文档的URLl”>

后面要学到框架struct 使用配置文件 ，使用外部的dtd

14、使用DTD来定义元素

语法:<!ELEMENT 元素名 约束>

简单元素：没有子元素的元素

<!ELEMENT name (#PCDATA)>

(#PCDATA) 约束name是字符串类型

EMPTY :元素为空（没有内容）

例如:DTD 代码

<!ELEMENT sex EMPTY>

Xml代码

<sex><sex>(元素为空)

ANY

DTD代码<!ELEMENT sex EMPTY>

<sex><sex>

复杂元素：

<!ELEMENT 元素名称（school，sex，name）>其中的元素在xml中只能出现一次

<!ELEMENT 元素名称（元素1，元素2，元素3）>

元素出现次数的限制

+：表示元素出现的次数大一1次

<!ELEMENT 元素名称（school+，sex，name）

\*：零次 一次或者多次

？：表示元素出现零次或者1次

子元素之间用逗号进行隔开

表示元素出现的顺序

15、使用dtd来定义属性

语法<!ATTLIST 元素名称 属性名称 属性类型 属性的约束> 一定要有空格

属性类型：CDATE：字符串

枚举类型：（aa|bb|cc）在一定的范围内出现值，但是只能出现一个

例如<!ELEMENT age (#PCDATE)>

<!ATTLIST age ID2 (AA|BB|CC) #REQUIRED> 必须出现的

Id：值只能是字母或者下划线开头”

<!ELEMENT age (#PCDATE)>

<!ATTLIST age ID3 ID #REQUIRED>

<age ID3=”2222”>111</age

属性的约束

#REQUIRED:属性必须存在

#IMPLIED:属性可有可无

#FIXED:表示固定的一个值 # FIXED “AAA”

属性的值必须是设定的值

直接值：直接写个值 “www” 标签的属性没有值是默认为 www 有值时使用设置的值

16、定义实体

语法：<!ENEIEY 实体名称 “实体的值”>

Dtd中<!ENTITY TEST “AHHAEHHE”>

使用实体：&实体名称

XML中<name>&TEST；</name>可以将hahahee引入进来

注：定义的实体需要写到内部的dtd里面，如果写到外部的dtd里面，某些浏览器得不到\*（指dtd引入方式）

# 17、xml解析的简介（写到java代码）

Xml:标记型文档

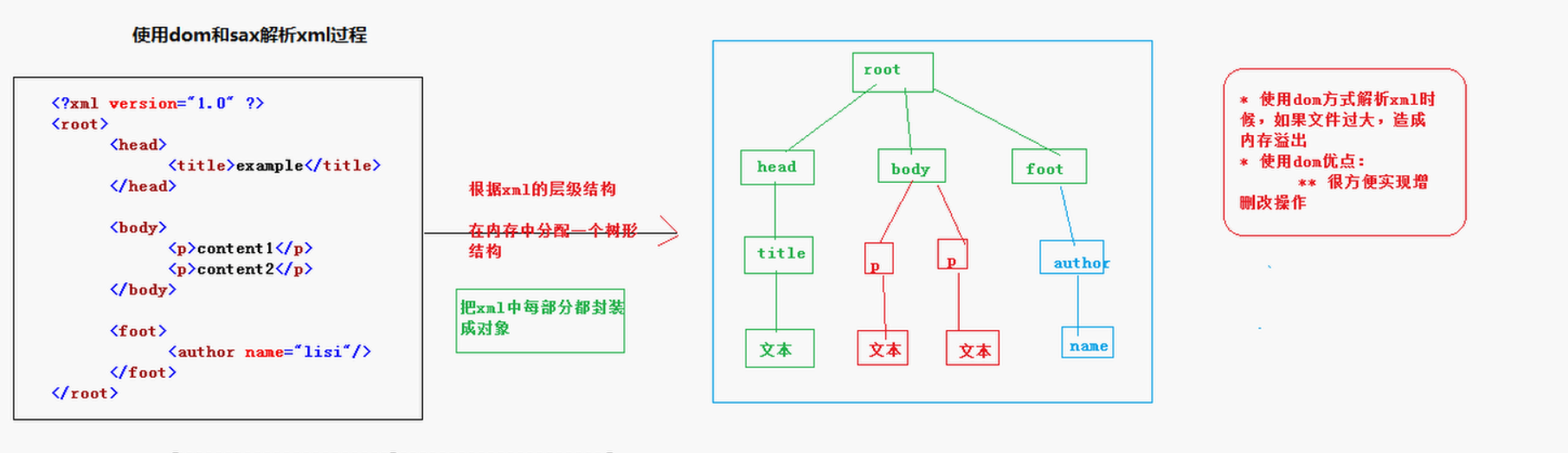
1Js使用dom解析标记型文档

根据html的层级结构，在内存中给他分配一个树形结构，将html的标签，属性，文本，都装成对象

Document 对象 element对象 属性对象 文本对象 NODE节点对象

Xml的解析方式（）

Dom解析





dom方式解析\*根据xm1的层级结构在内存中分配一个树形结构，把xm1的标签，属性和文本都封成对象

缺点：如果文件过大，造成内存溢出

优点：很方便实现增删改操作

sax方式解析

采用事件驱动，边读边解析

从上到下，一行一行的解析，解析到某一个对象，返回对象名称

缺点：不能实现增删改操作

优点：如果文件过大，不会造成内存溢出，方便实现查询操作

想要解析xml：首先需要解析器

不同公司和组织提供了针对dom和sax方式的解析器，通过api方式提供

Sun公司土工了针对dom和sax解析器 jaxp

Dom4j组织，针对dom和sax方式的解析器jaxp（解析器）（实际开发使用）

Jdom组织，针对dom和sax的解析器 jdom（解析器）

18、Jaxp的ap的查看i

Jaxp是javase的一部分

Jaxp解析器在jdk的javax.xml.parsers包里面

四个类：分别是针对dom和sax解析使用的类

Dom:

DoucmentBuilde:解析器类

这个类是个抽象类 不能实例化

此类的实例可以从DocumentBuilderFactory.newDocumentBuilder()方法获取到 一个方法 可以解析xml parse（“xml的路径”）返回是document的整个文档（在org.w3c.com中的）Document是一个接口，父接口是node 在document找不到想要的方法，在node中去找

在document中的方法

getElementByTagName（String tagname）

这个方法可以得到标签

返回集合NodeList

createElement(String tagName)

创建标签

createTextNode(String date)

创建文本

appendChild(Node new Child)

把文本添加到标签下面

removeChild(Node oldChild)

删除节点

getParentNode()

获取到父节点

NodeList中

Getlength()得到集合的长度

Item（int index） 下表取到具体的值

for（int i=0；i<list.getlength();i++）

{ list.item(i);

}

getTextContent();得到标签里面的内容

DocumentBuilderFactory 解析器工厂

也是抽象类

newInstance（）获取DocumentBuilderFactory的实力

Dax

SAXParser 解析器类

SAXParserFactory解析器工厂

19、使用jaxp实现查询操作

1、查询xml中所有的那么元素的值

//查询所有name元素的值

/\*

\*1、创建解析器工厂

DocumentBuilderFactory.newInstance()；

\*2、根据解析器工厂创建解析器

builderFactory.newDocumentBuilder()；

\*3、解析xm1返回document

\*Document document=builder.parse(\"src/person.xml\");

\*4、得到所有的name元素

使用document.getElemen[sByTagName(\"name\")；

\*5、返回集合，遍历集合，得到每一个name元素

-遍历getLength(）item()

-得到元素里面值使用getrextcontent()

\*\*/

\*\*月

**public** **static** **void** main(String[] args) **throws** ParserConfigurationException, SAXException, IOException {

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document= builder.parse("src/1.xml");

NodeList list = document.getElementsByTagName("name");

**for**(**int** i = 0;i<list.getLength();i++) {

Node name1=list.item(i);

String s = name1.getTextContent();

System.***out***.println(s);

}

获取某一个元素的值

\*1、创建解析器工厂

\*2、根据解析器工厂创建解析器

\*3、解析xm1，返回document

\*4、得到所有name元素

\*5、使用返回集合，里面方法item，下标获取具体的元素下表从零开始

\*6、得到具体的值，使用getrextcontent方法

DocumentBuilderFactory buildFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = buildFactory.newDocumentBuilder();

Document document = builder.parse("src/1.xml");

NodeList list = document.getElementsByTagName("name");

Node name1 = list.item(0);

String s1 = name1.getTextContent();

System.***out***.println(s1);

20、使用jaxp添加节点

在第一个P1的末尾添加nv

Xml

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

<person>

<p1>

<name>zhangsan</name>

<age>30</age>

</p1>

<p1>

<name>list</name>

<age>24</age>

</p1>

</person>

在第一p1下面添加 <sex>nv</sex>

\*1、创建解析器工厂

\*2、根据解析器工厂创建解析器

\*3、解析xm1，返回document

\*4、得到第一个p1

\*-得到所有p1，使用item方法下标得到

\*5、创建sex标签 createElement

\*6、创建文本 createTextNode

\*7、把文本添加到sex下面appendchild

\*8、把sex添加到第一个p1下面 appendchild

\*9、回写xml

\*\*/

**public** **static** **void** addSex() **throws** ParserConfigurationException, SAXException, IOException, TransformerException {

DocumentBuilderFactory buildFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = buildFactory.newDocumentBuilder();

Document document = builder.parse("src/1.xml");

NodeList list = document.getElementsByTagName("p1");

Node p1 = list.item(0);

Element sex1=document.createElement("sex");

Text text1 = document.createTextNode("nv");

sex1.appendChild(text1);

//回写写操作

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document),**new** StreamResult("src/1.xml"));

}

21、使用jaxp修改节点

修改第一个 p1下面的sex的内容是男

\*1、创建解析器工厂

\*2、根据解析器工厂创建解析器

\*3、解析xm1，返回document

\*4、得到sex item方法

\*5、修改sex里面的值setTextcontent方法

\*6、回写xm1

**public** **static** **void** modify() **throws** Exception {

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document = builder.parse("src/1.xml");

Node sex1= document.getElementsByTagName("sex").item(0);

sex1.setTextContent("nan");

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transFormer = transformerFactory.newTransformer();

transFormer.transform(**new** DOMSource(document), **new** StreamResult("src/1.xml"));

}

22、删除节点

步骤

1、创建解析器工厂

\*2、根据解析器工厂创建解析器

\*3、解析xm1，返回document

\*4、获取sex元素

\*5、获取sex的父节点使用getparentNode方法

\*6、删除使用父节点删除removechild方法

\*

\*7、回写xml

\*\*/

**public** **static** **void** delSex() **throws** Exception{

DocumentBuilderFactory builderFactory = DocumentBuilderFactory.*newInstance*();

DocumentBuilder builder = builderFactory.newDocumentBuilder();

Document document = builder.parse("src/1.xml");

//得到sex元素

Node sex1=document.getElementsByTagName("sex").item(0);

Node p1 = sex1.getParentNode();

p1.removeChild(sex1);

TransformerFactory transformerFactory = TransformerFactory.*newInstance*();

Transformer transformer = transformerFactory.newTransformer();

transformer.transform(**new** DOMSource(document),**new** StreamResult("src/1.xml") );

}

23、使用jaxp遍历节点

把xml中所有的元素名称都打印出来

遍历所有元素的名称、

\*1、创建解析器工厂

\*2、根据解析器工厂创建解析器

\*3、解析xm1，返回document

\*

\*====使用递归实现=====

\*4、得到根节点

\*5、得到根节点子节点

\*6、得到根节点子节点的子节点

**public** **static** **void** list1(Node node) {

//判断室元素类型是属性时打印否则会将空格和回车也打印因为解析是已经解析了空格和回车

**if**(node.getNodeType()==Node.***ELEMENT\_NODE***) {

System.***out***.println(node.getNodeName());

}

NodeList list = node.getChildNodes();//得到一子节点

//遍历list

**for**(**int** i = 0;i<list.getLength();i++) {

//得到一个节点

Node node1 = list.item(i);

//继续得到node1的子节点 递归

*list1*(node1);

}