**XML文件**

1. **用途：**数据存储和运输。
2. 表现：以“.xml”为文件扩展名的文件。
3. 存储结构：树形结构：根节点到子节点到子节点的结构。

4．作用：相同的xml文件把不同的软件联系起来（操作系统）。

5.java程序对于xml的四种解析方式：

DOM

SAX

DOM$J

JDOM

6.DOM方式解析xml文件：  
（1）创建一个DocumentBuilderFactory对象

（2）创建一个DocumentBuilder对象

（3）通过DocumentBuilder对象的parse(String filename)方法解析xml文件

Eg:

**public** **class** Dom {

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

// **TODO** Auto-generated method stub

//创建一个DocumentBuilderFactory对象

DocumentBuilderFactory d= DocumentBuilderFactory.*newInstance*();

//创建一个DocumentBuilder的对象

**try** {

DocumentBuilder d1=d.newDocumentBuilder();

//通过调用parse（）方法,将文件导入到当前项目下

Document d2=d1.parse("E:\\javawebProject\\jin.xml");

//通过方法.getElementsByTagName()；获得节点列表

NodeList list1=d2.getElementsByTagName("book");

//遍历每个book节点

System.***out***.println("一共有："+list1.getLength()+"本书");

**for**(**int** i=0;i<list1.getLength();++i)

{

//通过item(i)方法获取一个book节点

Node book=list1.item(i);//Node专门用来接收item(i)的返回值

NamedNodeMap n= book.getAttributes();//返回值是book里面所有的属性的集合

System.***out***.println("第"+(i+1)+"本书"+"共有"+n.getLength()+"个属性");

**for**(**int** b=0;b<n.getLength();++b)

{

//通过item()方法获取此时的属性

Node n1=n.item(b);

System.***out***.print("属性名："+n1.getNodeName()+" ");

System.***out***.println("属性值："+n1.getNodeValue()+" ");

}

/\*\*

\* 获取book节点的子节点

\*/

NodeList childNode=book.getChildNodes();

System.***out***.println("第"+(i+1)+"本书"+"一共有："+childNode.getLength()+"个子节点");

**for**(**int** g=0;g<childNode.getLength();++g)

{

Node childbook=childNode.item(g);

System.***out***.print("第"+(g+1)+"个节点的节点名为："+childbook.getNodeName()+" ");

//System.out.println("第"+(g+1)+"个节点的值为："+childbook.getFirstChild());

System.***out***.println("第"+(g+1)+"个节点的值为："+childbook.getTextContent());

}

System.***out***.println("结束第"+(i+1)+"一本书的遍历");

}

}

**catch** (SAXException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

**catch** (ParserConfigurationException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

//通过DocumentBuilder的parse(String Filename)解析文件

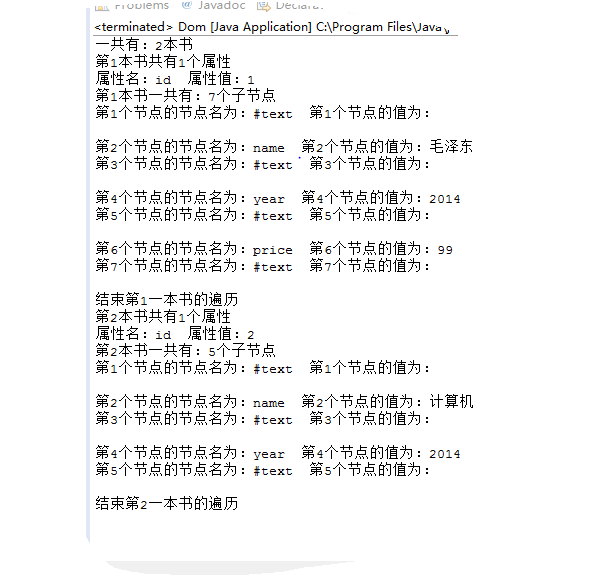
/\*\*

\* 注意在catch中书写语句

\*/

}

}



**文本内容：**

<?xml version="1.0" encoding="GBK"?>

<bookstore>

<book id="1">

<name><>毛泽东</name>

<year>2014</year>

<price>99</price>

</book>

<book id="2">

<name>计算机</name>

<year>2014</year>

</book>

</bookstore>

**7.SAX解析xml文件**

**(1)通过SAXPaserFactory的静态newInstance()的方法来获SAXPaserFactory的一个实例**

**(2)通过上面的实例获取SAXParser的实例。**

**(3)创建一个类，继承DefaultHandler,重写其中的一些方法进行业务处理，并创建这个类**

**public** **class** Book {

**private** String id;

**private** String name;

**private** String price;

**private** String year;

**public** String getId(){

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getPrice() {

**return** price;

}

**public** **void** setPrice(String price) {

**this**.price = price;

}

**public** String getYear() {

**return** year;

}

**public** **void** setYear(String year) {

**this**.year = year;

}

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

// **TODO** Auto-generated method stub

}

}

//

**public** **class** SAXPaserHandler **extends** DefaultHandler {

/\*\*

\* 用来遍历xml文件的开始标签

\*/

Book book=**null**;

String s=**null**;

**int** bookIndex=0;

**private** ArrayList<Book> a=**new** ArrayList<Book>();

**public** ArrayList<Book> getA() {

**return** a;

}

**public** **void** setA(ArrayList<Book> a) {

**this**.a = a;

}

@Override

**public** **void** startElement(String uri, String localName, String qName, Attributes attributes) **throws** SAXException {

//attributes：属性;attributes.getLength()：获取属性个数

**if**(qName.equals("book"))

{

//创建一个book对像

Book book=**new** Book();

bookIndex++;

**int** i=0;

**for**(;i<attributes.getLength();++i)

{

//获取属性名

System.***out***.print("book"+"元素的第"+(i+1)+"个属性名为："+attributes.getQName(i));

System.***out***.println("属性值为："+attributes.getValue(i));

**if**(attributes.getQName(i).equals("id"))

{

book.setId(attributes.getQName(i));

}

}

}

**else** **if**( !(qName.equals("book")&&!qName.equals("bookstory")))

{

//System.out.print("节点名是："+qName+" ");

}

// **TODO** Auto-generated method stub

**super**.startElement(uri, localName, qName, attributes);

}

/\*\*

\* 取得节点中的内容（值）

\*/

@Override

**public** **void** characters(**char**[] ch, **int** start, **int** length) **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.characters(ch, start, length);

s=**new** String(ch, start, length);

//if(!(s.trim().equals(" ")))

//{

//System.out.println(s);

//}

}

/\*\*

\* 用来遍历xml文件的结束标签

\*/

@Override

**public** **void** endElement(String uri, String localName, String qName) **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.endElement(uri, localName, qName);

//判断一本书是否遍历结束

**if**(qName.equals("book"))

{

a.add(book);

book=**null**;

System.***out***.println("结束第"+bookIndex+"本书的遍历");

}

**else** **if**(qName.equals("name"))

{

book.setName(s);

}

**else** **if**(qName.equals("year"))

{

book.setYear(s);

}

**else** **if**(qName.equals("price"))

{

book.setPrice(s);

}

}

/\*\*

\* 用来标识解析开始

\* **@param** args

\*/

@Override

**public** **void** startDocument() **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.startDocument();

System.***out***.println("SAX开始解析");

}//走到第一行的时候就会认为解析开始

/\*\*

\*用来标识解析结束

\* **@param** args

\*/

@Override

**public** **void** endDocument() **throws** SAXException {

// **TODO** Auto-generated method stub

**super**.endDocument();

System.***out***.println("SAX结束解析");

}//走到最后一行的时候就会认为解析结束

}

**public** **class** SAX {

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

// **TODO** Auto-generated method stub

//获取一个SAXPaserfactoey实例

SAXParserFactory f=SAXParserFactory.*newInstance*();

**try** {

//通过f获取SAXPaser的实例

SAXParser s=f.newSAXParser();

//创建SAXPaserHandler的对象

SAXPaserHandler s1=**new** SAXPaserHandler();

// for(Book book:s1.getA())

// {

// System.out.println(book.getId());

// System.out.println(book.getName());

// System.out.println(book.getPrice());

// System.out.println(book.getYear());

// }

**try** {

s.parse("jin.xml", s1);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

} **catch** (ParserConfigurationException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

} **catch** (SAXException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

