# XML 解析

主讲： 郭鑫

北京动力节点教育科技有限公司

动力节点课程讲义

DONGLIJIEDIANKECHENGJIANGYI

www.bjpowernode.com

# 第1章 XML 文件

## 1.1 什么是 xml 文件

（1） xml 是可扩展标识语言，(Extensible Markup Language)就是开发者在符合 xml 命名规则的基础之上，可以根据自己的需求定义自己的标签；

## 1.2 xml 文件的作用：主要是用来存储数据

## 1.3 解析 xml 文件的方法：DOM、DOM4J、SAX

# 第2章 Dom4J 解析 xml 文件

## 2.1 导入 Dom4J.jar 包

### 2.1.1 dom4j-1.6.1.jar

## 2.2 Dom4J 常用的对象：

### 2.2.1 SAXReader：读取 xml 文件到 Document 树结构文件对象

### 2.2.2 Document：是一个 xml 文档对象树，类比 Html 文档对象。

### 2.2.3 Element：元素节点。通过 Document 对象可以查找单个元素

## 2.3 Dom4j 解析步骤：

### 2.3.1 第一步：创建解析器

（1） SAXReader reader = new SAXReader();

### 2.3.2 第二步：Document 对象：通过解析器 read 方法获取

（1） Document doc = reader.read("studentInfo.xml");

### 2.3.3 第三步：获取 xml 根节点

（1） Element root = doc.getRootElement();

### 2.3.4 第四步：遍历解析子节点

## 2.4实例一：使用 Dom4j 解析 students.xml 文件

准备 students.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<students>
    <student>
        <name>吴飞</name>
        <college>java学院</college>
        <telephone>62354666</telephone>
        <notes>男,1982年生,硕士，现就读于北京邮电大学</notes>
    </student>
    <student>
        <name>李雪</name>
```

```
        <college>C++学院</college>
        <telephone>62358888</telephone>
        <notes>男,1987年生,硕士，现就读于中国农业大学</notes>
    </student>
    <student>
        <name>Jack</name>
        <college>PHP学院</college>
        <telephone>66666666</telephone>
        <notes>我是澳洲人</notes>
    </student>
    <student>
        <name>Lucy</name>
        <college>Android学院</college>
        <telephone>88888888</telephone>
        <notes>我是美国人</notes>
    </student>
</students>
```

### 2.4.1 第一步：创建解析器

```
SAXReader   reader = new SAXReader();
```

### 2.4.2 第二步：通过 SAXReader 的 read 方法获取 Document 对象

```
Document doc = reader.read("students.xml");
```

### 2.4.3 第三步：获取 xml 的根节点

```
Element root = doc.getRootElement();
```

### 2.4.4 第四步：通过 Iterator 遍历根节点 root 获取子节点 student

```
//遍历 root 根节点下的 student 子节点
for(Iterator iter =   root.elementIterator();iter.hasNext();){
//获取 student 节点对象
Element stuElement = (Element)iter.next();
//遍历 stuElement 节点下的所有子节点：name,colleage,telephone,note
for(Iterator innerIter = stuElement.elementIterator();innerIter.hasNext();){
//获取 student 节点下的子节点对象
Element innerElement = (Element)innerIter.next();
//通过 innerElement 的 getName()获取节点名称，getStringValue()获取节点值
}
```

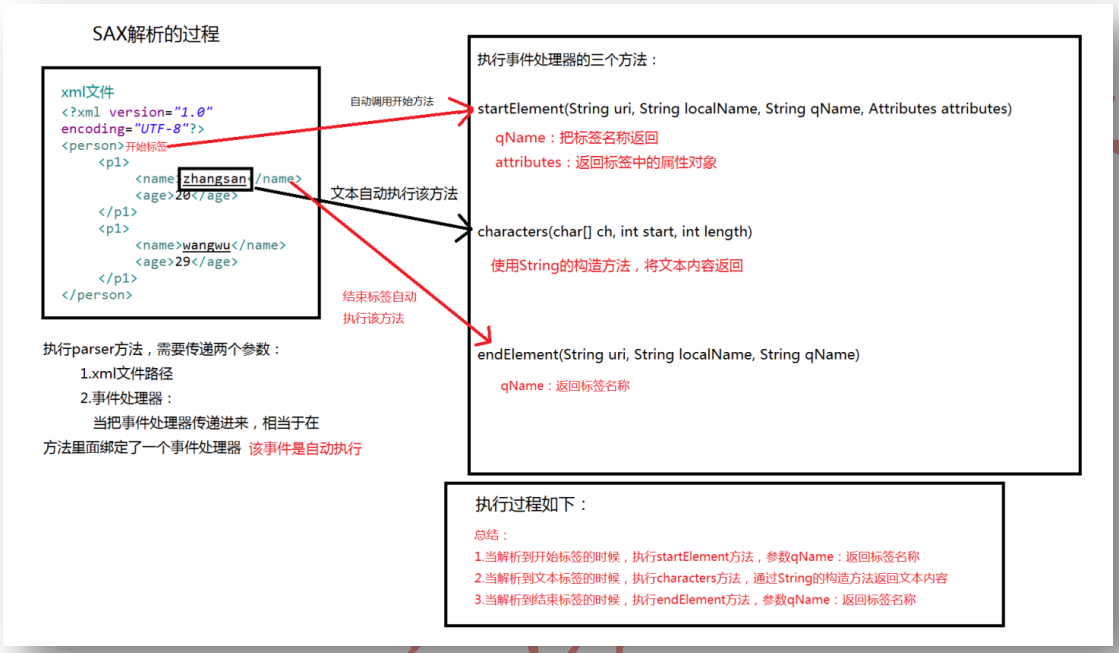
}

```java
package com.bjpowernode.dom4jparser;

import java.util.Iterator;

import org.dom4j.Document;
import org.dom4j.Element;
import org.dom4j.io.SAXReader;

public class Dom4jTest01 {

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

        //创建解析器
        SAXReader reader = new SAXReader();

        //通过SAXReader的read方法获取Document对象
        Document doc = reader.read("students.xml");

        //获取xml文件的根节点
        Element root = doc.getRootElement();

        //遍历根节点获取子studentElement子节点
        for(Iterator iter = root.elementIterator(); iter.hasNext();){
            //获取所student节点对象
            Element stuElement = (Element) iter.next();

            //打印出student节点名称
            System.out.println(stuElement.getName()+"信息如下：");

            //遍历student节点下的子节点：name、college、telephone、note
            for(Iterator innerIter = stuElement.elementIterator();innerIter.hasNext();){
                //获取student子节点对象
                Element innerElement = (Element) innerIter.next();


                String innerElementName = innerElement.getName();
                String innerElementValue = innerElement.getStringValue();
                System.out.println(innerElementName+":"+innerElementValue);


            }

        System.out.println("----------------------------");
        }


    }

}
```

# 第3章 Sax 解析 xml 文件

## 3.1 SAX 方式：事件驱动，边读边写

### 3.1.1 执行过程如下图：



### 3.1.2 优点：无需将整个文档加载到内存中,所以内存消耗少，适合解析特别大的 xml 文件

### 3.1.3 SAX 解析四步曲：

（1） 创建解析工厂：通过 newInstance()方法获取

SAXParserFactory saxParserFactory = SAXParaserFactory.newInstance();

```
//1.创建解析器工厂
SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
```

（2） 创建解析器

SAXParser saxParser = saxParserFactory.newSAXParser();

```
//2.创建解析器
SAXParser saxParser = saxParserFactory.newSAXParser();
```

（3） 执行 parser 方法，传入两个参数：xml 文件路径、事件处理器

saxParser.parser(“person.xml”, new MyDefaultHander1());

```
//3.执行parser方法
saxParser.parse("person.xml", new MyDefaultHander1());
```

## （4） 创建一个类，继承 DefaultHander 类，重写三个方法：

**A、 startElement　　获取开始标签，重要的两个参数说明**

**a、 qName：把标签名称返回**

**b、 attributes：返回标签中的属性对象**

**B、 character　　获取标签文本内容**

**C、 endElement　　获取结束标签**

```
/**
 *  实现处理器，需要继承DefaultHander类
 * @author Administrator
 *
 */
class MyDefaultHander1 extends DefaultHandler{

    @Override
    public void startElement(String uri, String localName, String qName,
            Attributes attributes) throws SAXException {
        System.out.print("<"+qName+">");
    }

    @Override
    public void characters(char[] ch, int start, int length)
            throws SAXException {
        System.out.print(new String(ch,start,length));
    }

    @Override
    public void endElement(String uri, String localName, String qName)
            throws SAXException {
        System.out.print("</"+qName+">");
    }

}
```

### 3.1.4 实例一：获取 person.xml 文件并且原样打印出

**准备：person.xml 文件**

```
<?xml version="1.0" encoding="UTF-8"?>
<person>
    <p1>
        <name>zhangsan</name>
        <age>20</age>
    </p1>
    <p1>
        <name>wangwu</name>
        <age>29</age>
    </p1>
</person>
```

### （1） 创建 SAXParser 解析工厂对象

```
SAXParserFactory saxParserFactory = SAXParserFactory.newInstance();
```

### （2） 创建解析器

```
SAXParser saxParser = saxParserFactory.newSAXParser();
```

### （3） 执行 parser 方法

```
saxParser.parser("person.xml",new MyDefaultHander());
```

### （4） 创建 MyDefaultHander 类继承 DefaultHander 类

A、 startElement 方法中，最重要的两个参数

a、 qName：把开始标签名称返回

b、 attributes：返回标签中的属性对象

c、 --System.out.print("<"+qName+">");

B、 characters(char[] ch, int start, int length)方法：获取标签文本内容

a、 通过 String 的构造方法 new String(ch,start,length);构造出一个 String 字段串

b、 --System.out.print(new String(ch,start,length));

C、 endElement 方法中，最重要的一个参数

a、 qName：把结束标签名称返回

b、 --System.out.print("</"+qName+">");

```java
//4.创建一个类，继承DefaultHander类，重写三个方法
/**
 * 实现处理器，需要继承DefaultHander类
 * @author Administrator
 *
 */
class MyDefaultHander1 extends DefaultHandler{

    @Override
    public void startElement(String uri, String localName, String qName,
            Attributes attributes) throws SAXException {
        System.out.print("<"+qName+">");
    }

    @Override
    public void characters(char[] ch, int start, int length)
            throws SAXException {
        System.out.print(new String(ch,start,length));
    }

    @Override
    public void endElement(String uri, String localName, String qName)
            throws SAXException {
        System.out.print("</"+qName+">");
    }
```

# 第4章 使用 Dom4j 的 xPath 解析 xml 文件

## 4.1 XPath 语法

### 4.1.1 官方语法地址：http://www.w3school.com.cn/xpath/index.asp

### 4.1.2 xpath 使用路径表达式来选取 XML 文档中的节点或节点集。节点是通过沿着路径 (path) 或者步 (steps) 来选取的

## 4.2 xpath 语法选取节点

### 4.2.1 bookstore.xml 文件

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

```
<bookstore>
   <book>
      <title lang="eng">Harry Potter</title>
      <price>29.99</price>
   </book>
   <book>
      <title lang="eng">Learning XML</title>
      <price>39.95</price>
   </book>
</bookstore>
```

4.2.2 选择节点：XPath 使用路径表达式在 XML 文档中选取节点。节点是通过沿着路径或者 step 来选取的。下面列出了最有用的路径表达式：

| 表达式 | 描述 |
| --- | --- |
| nodename | 选取此节点的所有子节点。 |
| / | 从根节点选取。 |
| // | 从匹配选择的当前节点选择文档中的节点，而不考虑它们的位置。 |
| . | 选取当前节点。 |
| .. | 选取当前节点的父节点。 |
| @ | 选取属性。 |

### 4.2.3 实例：在下面的表格中，我们已列出了一些路径表达式以及表达式的结果：

| 路径表达式 | 结果 |
| --- | --- |
| bookstore | 选取 bookstore 元素的所有子节点。 |
| /bookstore | 选取根元素 bookstore。<br>注释：假如路径起始于正斜杠( / )，则此路径始终代表到某元素的绝对路径！ |
| bookstore/book | 选取属于 bookstore 的子元素的所有 book 元素。 |
| //book | 选取所有 book 子元素，而不管它们在文档中的位置。 |
| bookstore//book | 选择属于 bookstore 元素的后代的所有 book 元素，而不管它们位于 bookstore 之下的什么位置。 |
| //@lang | 选取名为 lang 的所有属性。 |

### 4.2.4 text()获取节点的文本内容函数

## 4.3示例一：获取 sys-config.xml 文件的配置信息

**准备工作：**

**（1） 导入 dom4j-1.6.1.jar 和 jaxen-1.1-beta-7.jar**

**（2） sys-config.xml 文档**

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
   <database-info>
```

```
    <driver-name>com.mysql.jdbc.Driver</driver-name>

  <url>jdbc:mysql://192.168.1.151:8080/bjpowernode</url>
    <user>root</user>
    <password>123</password>
  </database-info>
</config>
```

### 4.3.2 第一步：通过 SAXReader 获取解析器

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

### 4.3.3 第二步：通过解析器的 read 方法获取 Document 对象

**Document doc = reader.read("sys-config.xml");**

### 4.3.4 第三步：通过 xpath 语法获取 driver-name 节点

**Element driverNameElt =**

**doc.selectObject("/config/database-info/driver-name");**

**String driverName= driverNameElt.getStringValue();**

## 4.4示例二：解析 server.xml 文件:获取端口号

**准备 server.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<server>
   <service>
      <connector port="8080"></connector>
   </service>
</server>
```

#### 4.4.1 第一步：创建 SAXReader 解析器

```
SAXReader reader = new SAXReader();
```

#### 4.4.2 第二步：通过解析器 read 方法获取 Document 对象

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

#### 4.4.3 第三步：通过 xpath 语法获取 connector 节点

```
Element connectorElt =
document.selectSingleNode("/server/service/connector");
```

#### 4.4.4 第四步：获取 connector 节点的 port 属性对象

```
Attribute portAttr = connector.attribute("port");
String   portStr = portAttr.getStringValue();
```

### 4.5示例三：解析 books.xml 文件(org.w3c.dom.Document)

准备 books.xml 文件

```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
   <book category="children">
      <title lang="en">Harry Potter</title>
      <author>J K. Rowling</author>
```

```
    <year>2005</year>
    <price>29.99</price>
</book>

<book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
</book>

<book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
</book>

<book category="web">
    <title lang="uk">XQuery Kick Start</title>
    <author>James McGovern</author>
    <year>2003</year>
```

```
    <price>49.99</price>
  </book>

</bookstore>
```

**4.5.1 第一步：通过 DocumentBuilderFactory 获取解析工厂**

**DocumentBuilderFactory builderFactory =**

**DocumentBuilderFactory.newInstance();**

**4.5.2 第二步：获取解析器**

**DocumentBuilder builder = builderFactory.newDocumentBuilder();**

**4.5.3 第三步：通过解析的 parser 方法获取 Document 对象**

**Document document = builder.parse(new File("books.xml"));**

**4.5.4 第四步：获取 XPath 对象**

**XPath xPath = XPathFactory.newInstance().newXPath();**

**4.5.5 //获取 bookstore 节点下 book 属性 category 值为 web 下的第二个 title 节点的文本内容**

**//获取 bookstore 节点下 book 属性 category 值为 web 下的第二个 title 节点的文本内容**

```
String webTitle = (String) xPath.evaluate(
        "/bookstore/book[@category='web'][2]/title/text()", document,
        XPathConstants.STRING);
System.out.println("/bookstore/book[@category='web'][2]/title="+webTitle);
```

```
xPath.evaluate("/bookstore/book[@category='web'][2]/title/
text()",document, XPathConstants.STRING);
```

4.5.6 //获取 bookstore 节点下 book 属性 category 值为 web 的 titile 属性为 en 的节点内容

//获取 bookstore 节点下 book 属性 category 值为 web 的 titile 属性为 en 的节点内容

```
String titleEn = (String) xPath
        .evaluate("/bookstore/book[@category='web']/title[@lang='en']/text()",
                document, XPathConstants.STRING);
```

```
xPath.evaluate("/bookstore/book[@category='web']/title[@lang='en']/text()",document, XPathConstants.STRING)
```

4.5.7 //获取 bookstore 下 book 属性 category 值为 cooking 的 title 的 lang 属性的值

```
//获取bookstore下book属性category值为cooking的title的lang属性的值

xPath.evalute("/bookstore/book[@category='cooking']/title/@lang",document, XPathConstants.STRING);
```

4.5.8 // 获取 bookstore 节点下所有 book 的节点集合

```
// 获取 bookstore 节点下所有 book 的节点集合

NodeList books = (NodeList)xPath.evaluate("/bookstore/book",document, XPathConstants.NODESET);

//遍历 books 节点

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

//获取 book 节点

Node book = books.item(i);

//获取 book 标签子节点 title 的文本内容

xPath.evaluate("title/text()",book, XPathConstants.STRING);
```

```
}
```